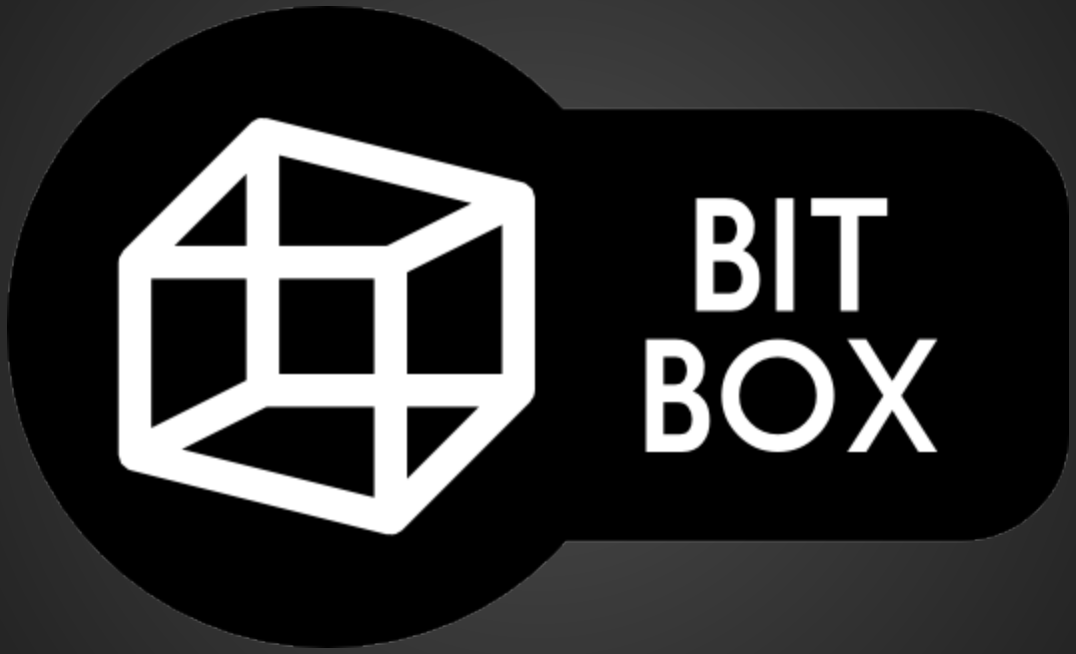

BIT
BOX

plan

0 - Introduction

- goal
- general presentation

1 - Hardware

2 - Software

0 - Intro

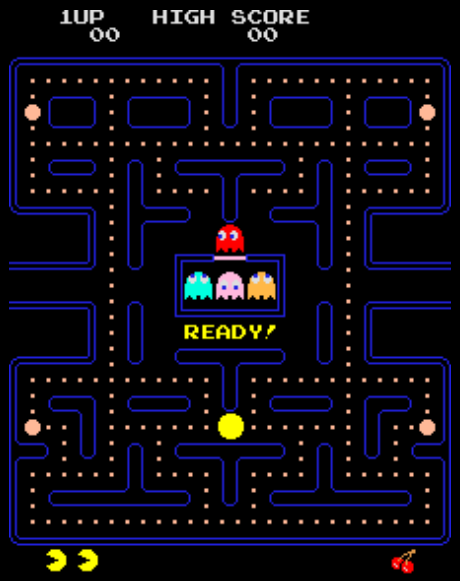
goal

a modern, DIY, simple videogame console

Retrogaming

Oldskool : nice, simple & fun games

DIY ⇒ low spec ⇒ retrogaming ⇒ ♥

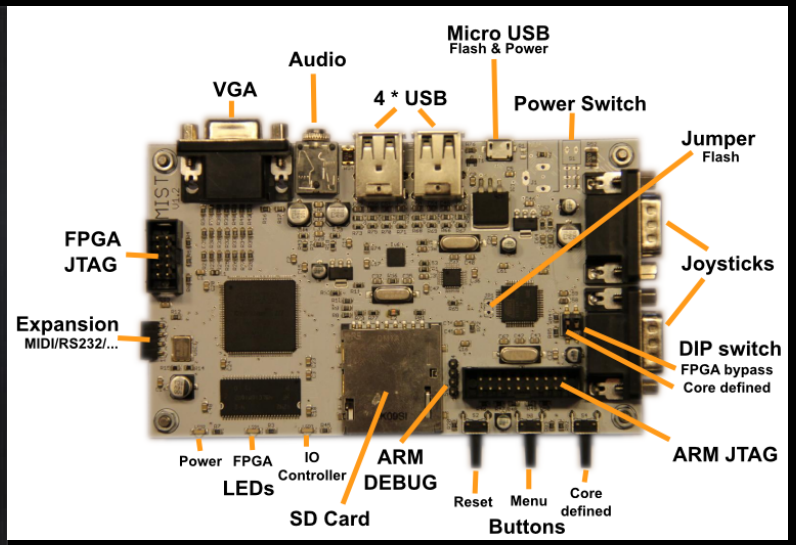


Emulation

Real hardware : Old, hard to find, strange to use

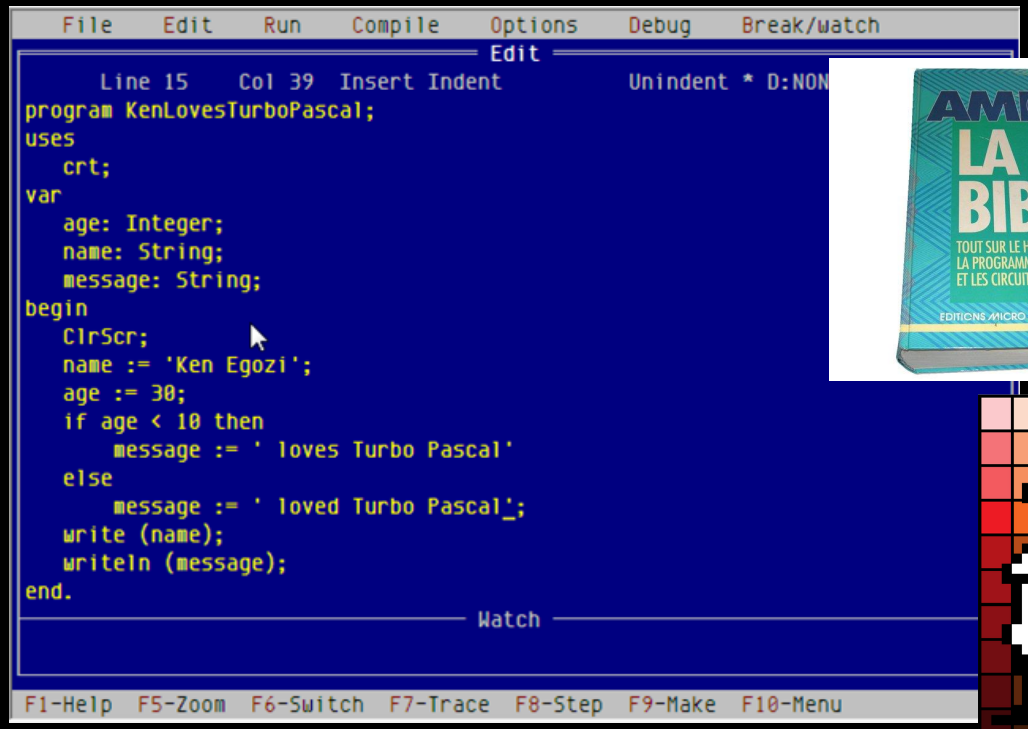
Software emulator : not the real deal.

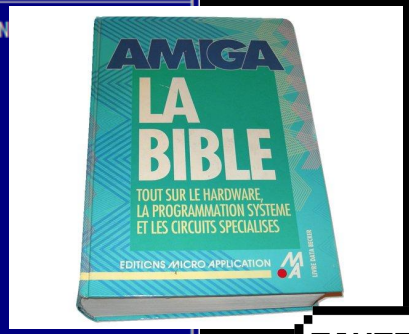
Hardware emulator : Wizardry-level hacking … used to emulate BUGS !



Retro Coding

The game is in the code ! No OS, raw “speed” (no frameworks !)





Goals : 16bits, Free, Open & DIY

- Target : 16-bit era games
- Big Screen (TV)
- As simple as possible

- OPEN / FOSS
  - Low level coding
  - OPEN toolchain ! (gcc/gdb)
  - NO binary blob / driver ! (hello raspi !)

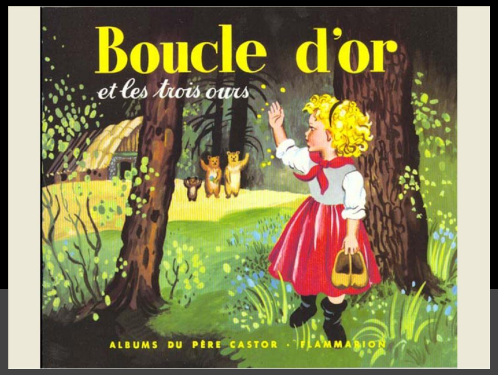
Goals : DIY Hardware

- One standard chip
  - no FPGA (expensive)
  - no external RAM / Flash
  - as cheap as possible
  - simply sourceable

- Modern interfaces
  - Off the shelf peripherals

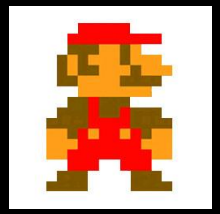
- Simple to manufacture
  - OK for double sided PCB and CMS (TQFP OK)
  - Doable by an individual in his basement
  - hand-soldering (no BGA)

Consoles



	XBone	attiny pong
Specs	8 core cpu 1.7GHz 8 GB ram	ATTiny 2313 20MHz 128 **B ram** 2k flash
Games	Bluray 50GB	Pong !
Screen	HDMI GPU 12 cores 1080p to 4K	B&W TV
Sound	...	beep
Open / Hackable	(drm)	open (limited!)

8bits

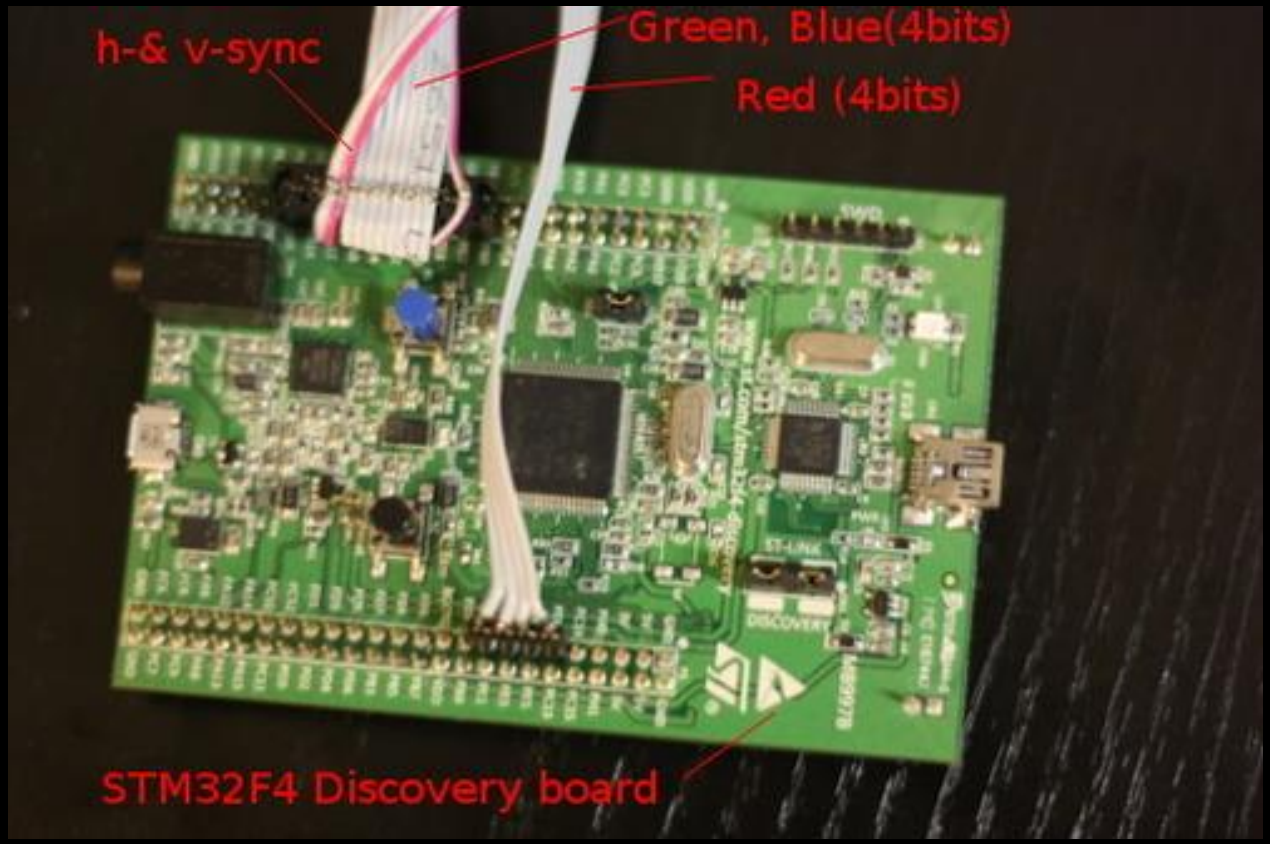


	NES	UzeBox
Specs	8bit 6502 1.79 MHz 2KB RAM	AVR 644 28MHz 4KB RAM
Games	cartridges 32k-1M (+ram?)	SD Card/ 64k Flash
Screen	HW Tiles/ Sprites RF output 256x224 16/64 colors	Software (no GPU) + RGB to composite chip 240x224 256 C total (rgb 323)
Sound	5 voices (1 with 4bit samples)	4x 8bits@15kHz
Open / Hackable	old, closed but very well documented	(completely open)

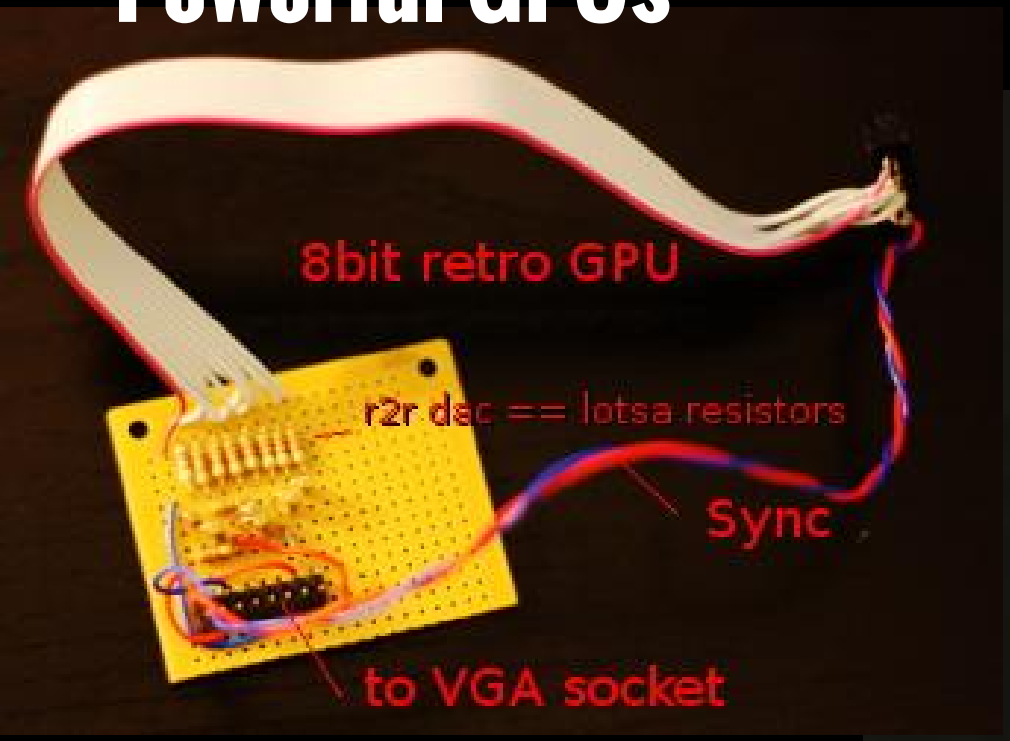
16bits

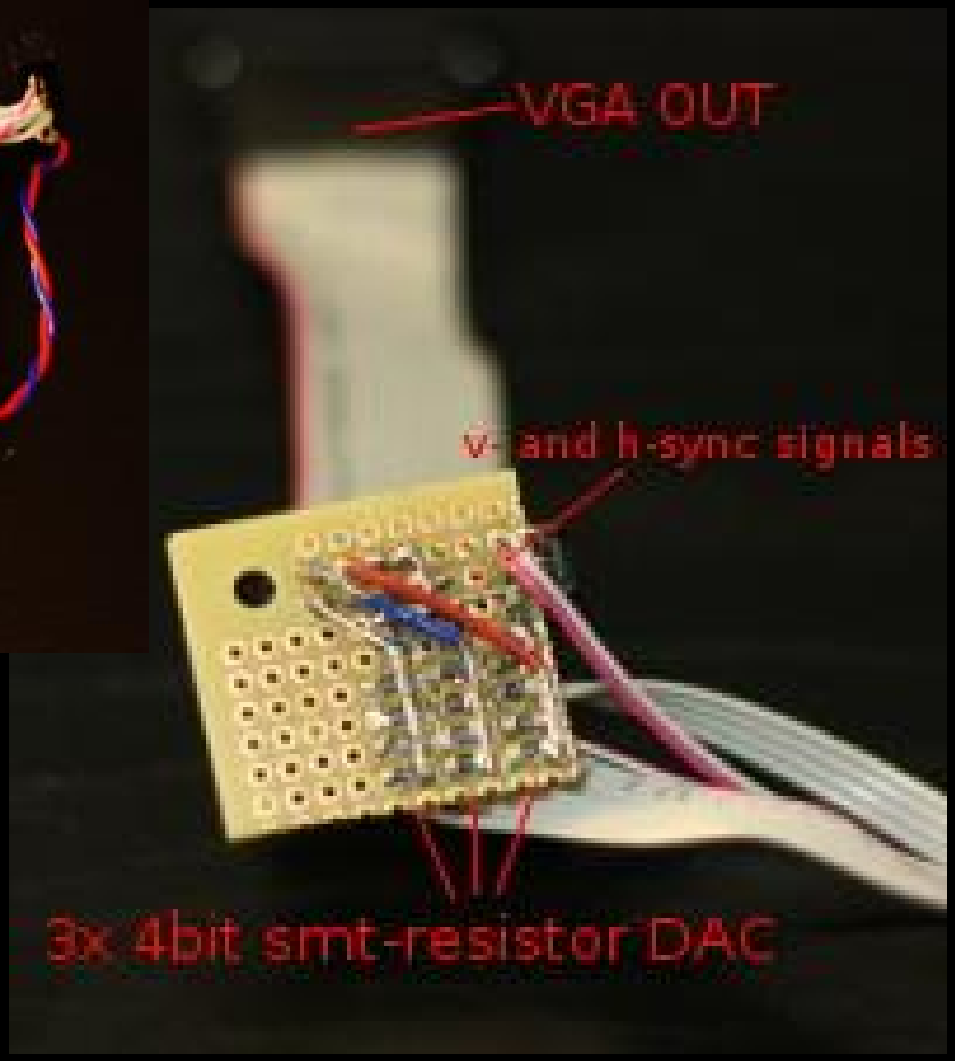
	SNES	Bitbox	raspberry PI
Specs	5A22 16bits 3.5MHz 192k RAM	STM32 ARM 32bits 168MHz 192k RAM	700MHz SoC ARMv6 512M RAM
Games	Cartridges 1-8 MB	uSD 1MB Flash	uSD
Screen	composite/S-video GPU sprites & tiles, 240x224 (max 512x448) 15bpp	VGA software 640x480 320x200 -> 800x600 15bpp	HDMI 3D Videocore IV GPU
Sound	St 16 bits, 32kHz 16 voices, ADPCM,	Stereo 12bits 32+ kHz	48kHz 16bits
Open / Hackable	closed but well documented	^_^	binary blob, chip not for sale, complex hw

Developing on the cheap



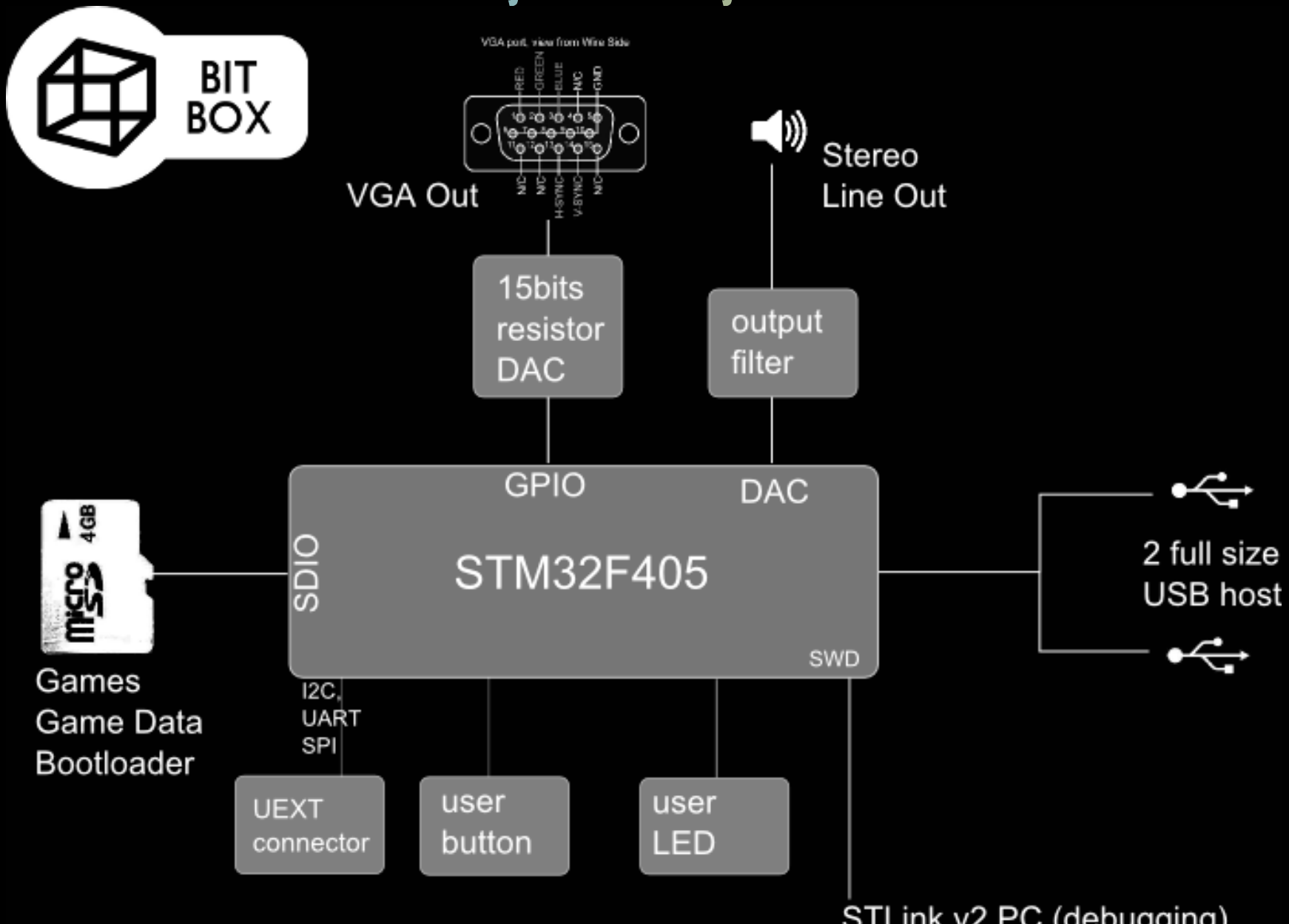
Powerful GPUs



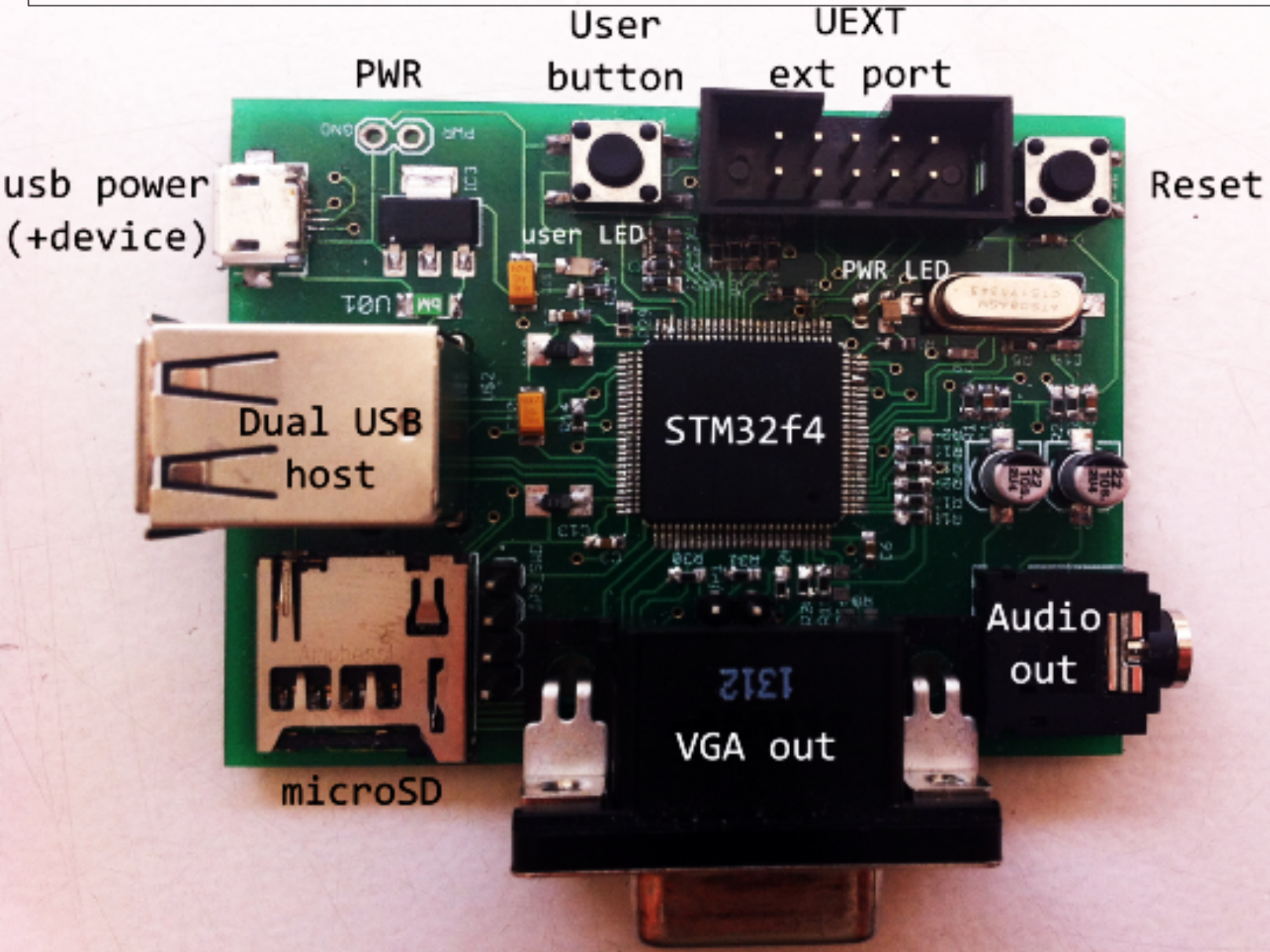


finally ...

Bitbox rev2 : USB, Stereo, 15Bit color



Bitbox rev2


User
button
UEXT
ext port
PWR
usb power
(+device)
Reset
user LED
PWR LED
Dual USB
host
STM32f4
Audio
out
VGA out
microSD

1 - Hardware

What's in a Game Console ?

- CPU / RAM / Flash : Microcontroller

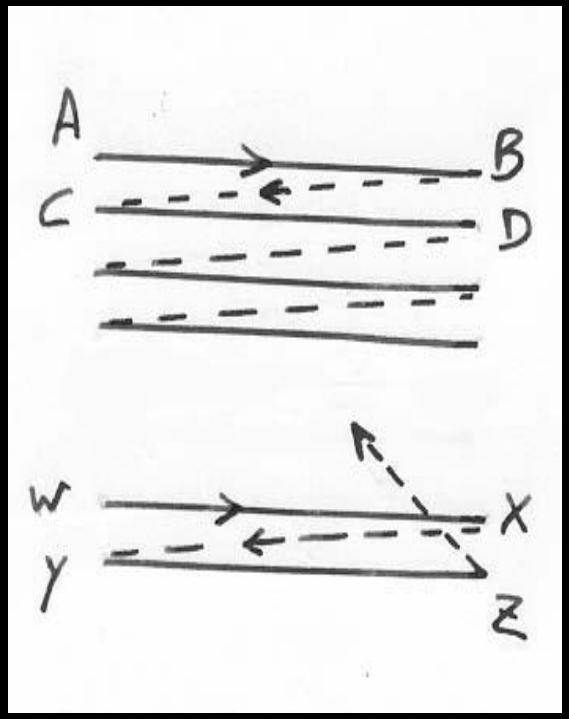
- Video
- Sound
- Input peripherals
- Mass storage
- Game transfer

Choosing a Microcontroller (CPU)

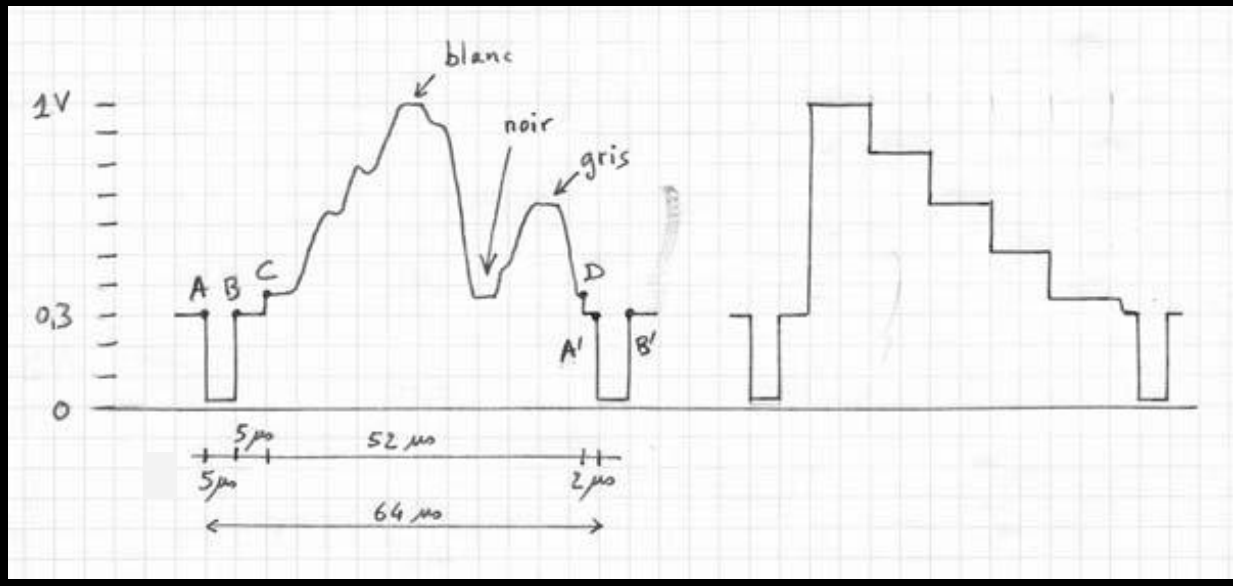
- Cortex-M (ARM 32bits)
- STM32F4 : recent generation
  - 1MB Flash + accelerator , 192 KB ram
  - 168 MHz (+O/C ..)
  - stm32f405 : 7€ / 10k (remember : RPi=35€)
  - peripherals : native USB host / SDIO :)

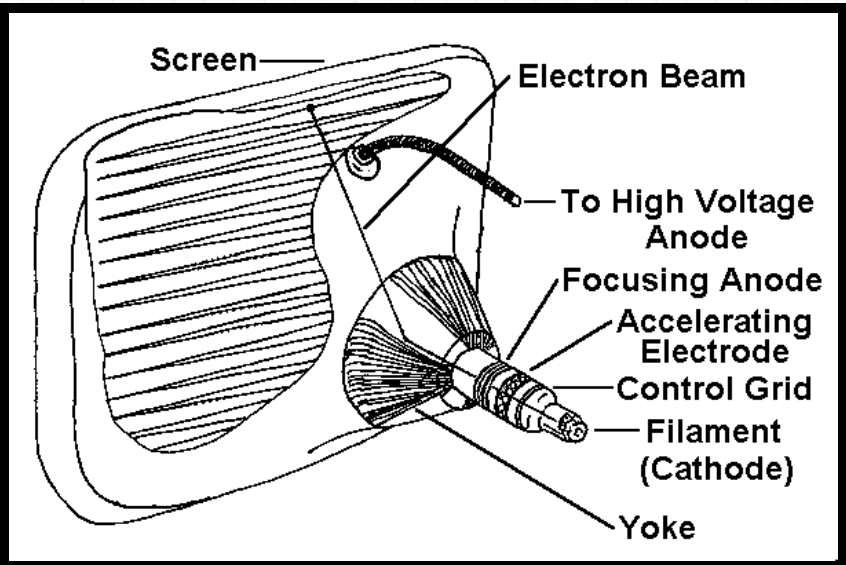
- Prototype board STM32F4Discovery
  - very cheap, debug interface SWD
  - non-free toolchains :(but GCC can be made to work

Video signal : B&W

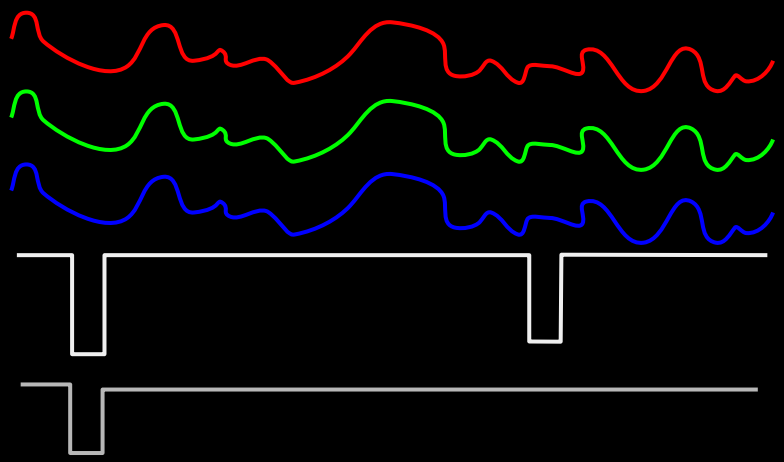


source : http://f5ad.free.fr/ATV-QSP_F5AD_Le_signal_video.htm

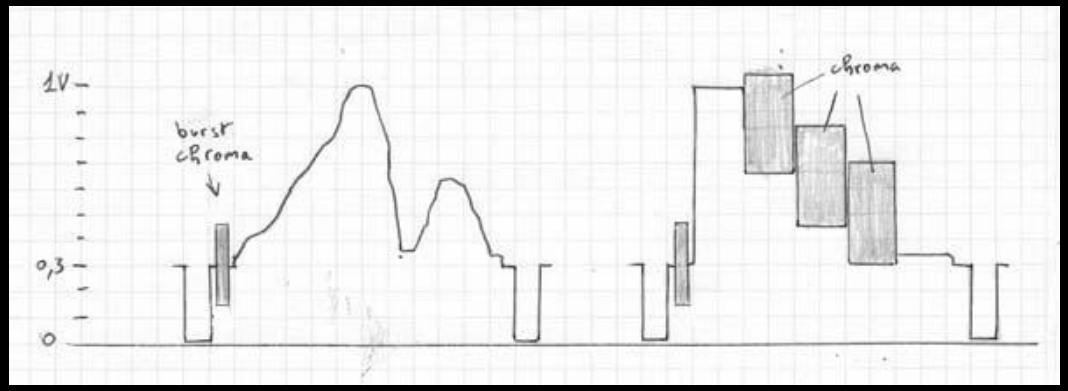




Video Signals : color

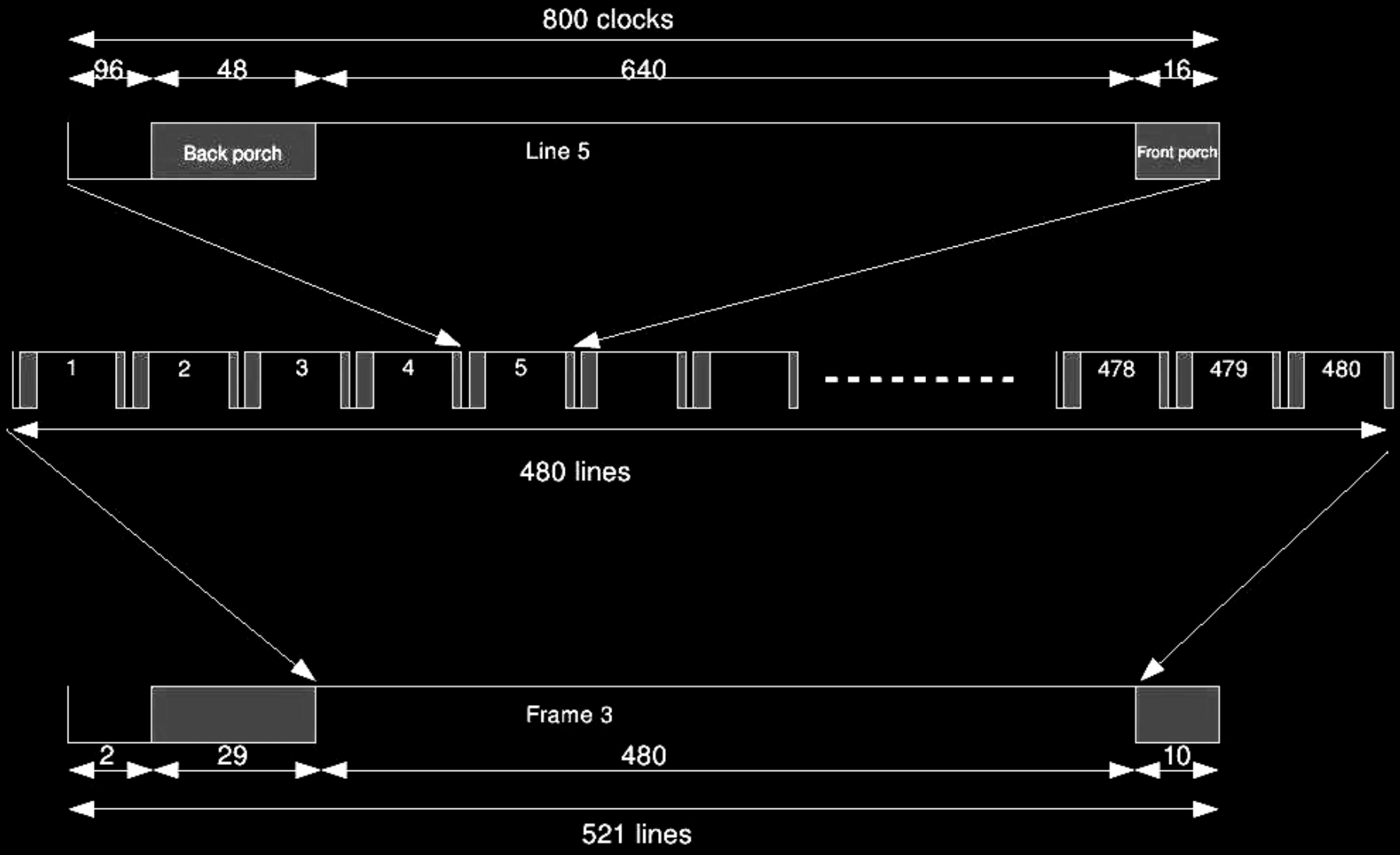


VGA (5 wires)
Component
YPbPr (3 wires)



PAL / SECAM / NTSC
SVideo : Luma/
Chroma (2fils)
Composite (1fil)

VGA Timings


800 clocks
96
48
640
16
Back porch
Line 5
Front porch
1
2
3
4
5
478
479
480
480 lines
Frame 3
2
29
480
10
521 lines

VGA Timings

- Standard VGA 640x480 60fps
  - Line clock : 60*(480+41) = 31260 Hz
  - Pixel clock : 31.2kHz * 800 = 25MHz

or other combinations, standard or not !

(modelines X, “multisync” …)

Hardware : stm32f4 key features

Key Features

- Core: ARM 32-bit Cortex™-M4 CPU with FPU, Adaptive real-time accelerator (ART Accelerator™) allowing 0-wait state execution from Flash memory, frequency up to 168 MHz, memory protection unit, 210 DMIPS/1.25 DMIPS/MHz (Dhrystone 2.1), and DSP instructions
- Memories
- Up to 1 Mbyte of Flash memory
- Up to 192+4 Kbytes of SRAM including 64-Kbyte of CCM (core coupled memory) data RAM
- Flexible static memory controller supporting Compact Flash, SRAM, PSRAM, NOR and NAND memories
- LCD parallel interface, 8080/6800 modes
- Clock, reset and supply management
- 1.8 V to 3.6 V application supply and I/Os
- POR, PDR, PVD and BOR
- 4-to-26 MHz crystal oscillator
- Internal 16 MHz factory-trimmed RC (1% accuracy)
- 32 kHz oscillator for RTC with calibration
- Internal 32 kHz RC with calibration
- Sleep, Stop and Standby modes
- V_{BAT} supply for RTC, 20×32 bit backup registers + optional 4 KB backup SRAM
- 3×12-bit, 2.4 MSPS A/D converters: up to 24 channels and 7.2 MSPS in triple interleaved mode
- 2×12-bit D/A converters
- General-purpose DMA: 16-stream DMA controller with FIFOs and burst support
- Up to 17 timers: up to twelve 16-bit and two 32-bit timers up to 168 MHz, each with up to 4 IC/OC/PWM or pulse counter and quadrature (incremental) encoder input
- Debug mode Serial wire debug (SWD) & JTAG interfaces, Cortex-M4 Embedded Trace Macrocell™
- Up to 140 I/O ports with interrupt capability, Up to 136 fast I/Os up to 84 MHz
- Up to 15 communication interfaces
- Up to 3 × I^2C interfaces (SMBus/PMBus)
- Up to 4 USARTs/2 UARTs (10.5 Mbit/s, ISO 7816 interface, LIN, IrDA, modem control)
- Up to 3 SPIs (42 Mbits/s), 2 with muxed full-duplex I2S to achieve audio class accuracy via internal audio PLL or external clock
- 2 × CAN interfaces (2.0B Active)
- SDIO interface
- Advanced connectivity
- USB 2.0 full-speed device/host/OTG controller with on-chip PHY
- USB 2.0 high-speed/full-speed device/host/OTG controller with dedicated DMA, on-chip full-speed PHY and ULPI
- 10/100 Ethernet MAC with dedicated DMA: supports IEEE 1588v2 hardware, MII/RMII
- 8- to 14-bit parallel camera interface up to 54 Mbytes/s
- True random number generator
- CRC calculation unit
- 96-bit unique ID
- RTC: subsecond accuracy, hardware calendar

MCU usage

MCU + Memory : game + video gen

GPIO : input/output @ 25MHz, PWM !

DMA : memory→ 15 bit GPIO, no gpu

Timers/counters : pixel clk, Vsync, DMA triggers, PWM GPIO, audio DMA

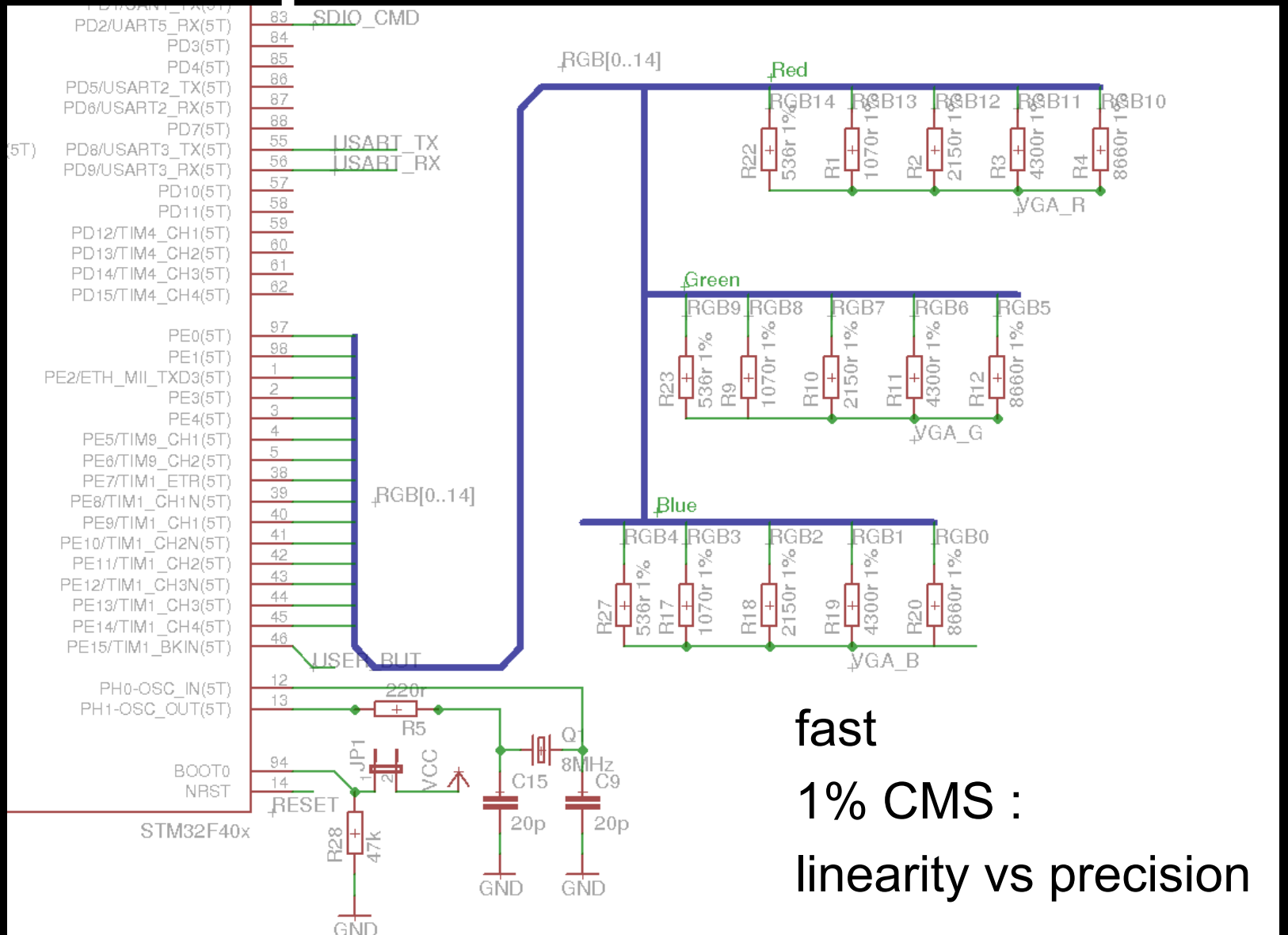
DAC : audio

USB, SDIO

ADC, RTC, SPI, I2C, UART, …

Clocks : USB/SDIO 48MHz, 168MHz, …

Video output : resistor DAC



fast

1% CMS :

linearity vs precision

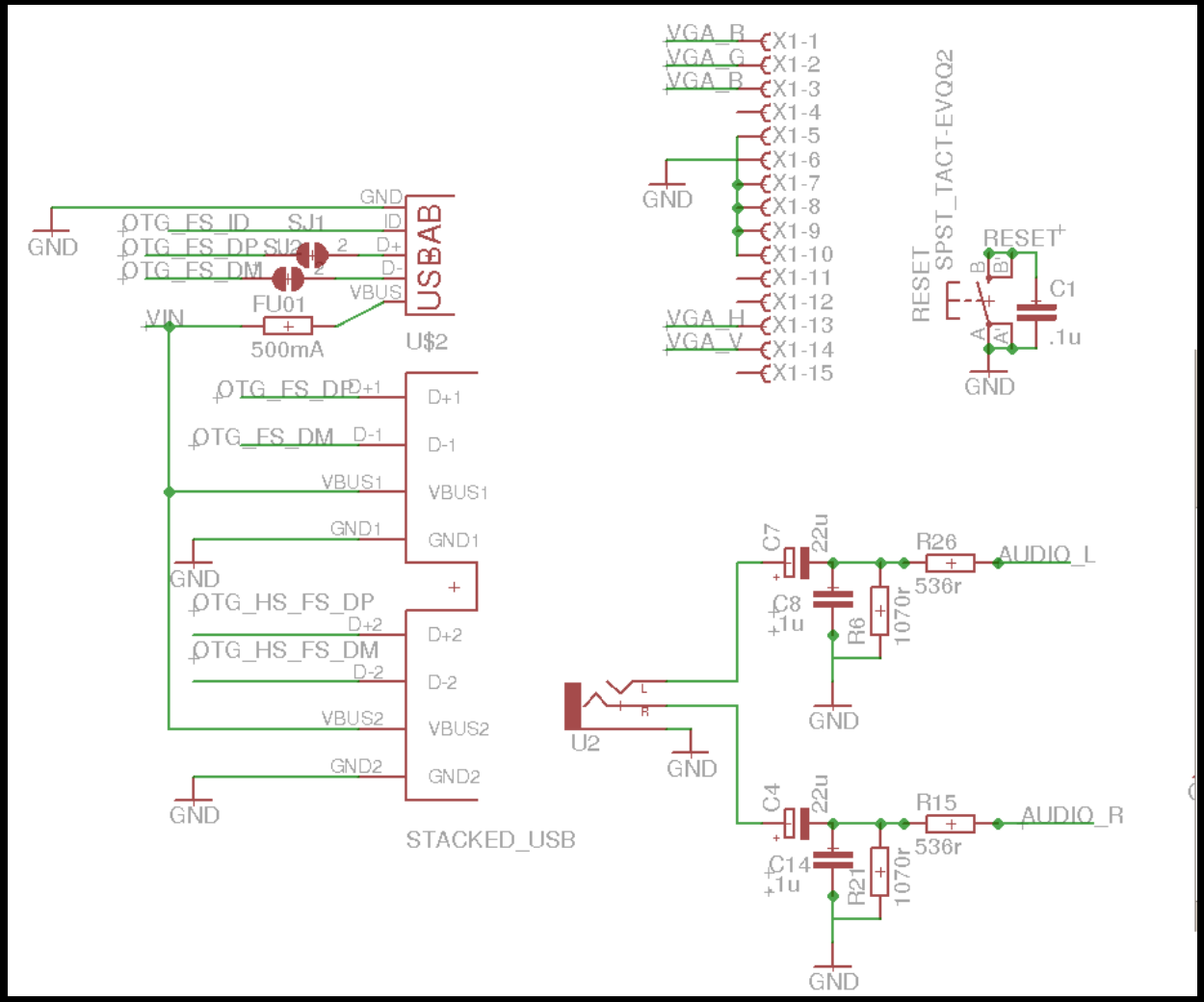
Sound

Sound

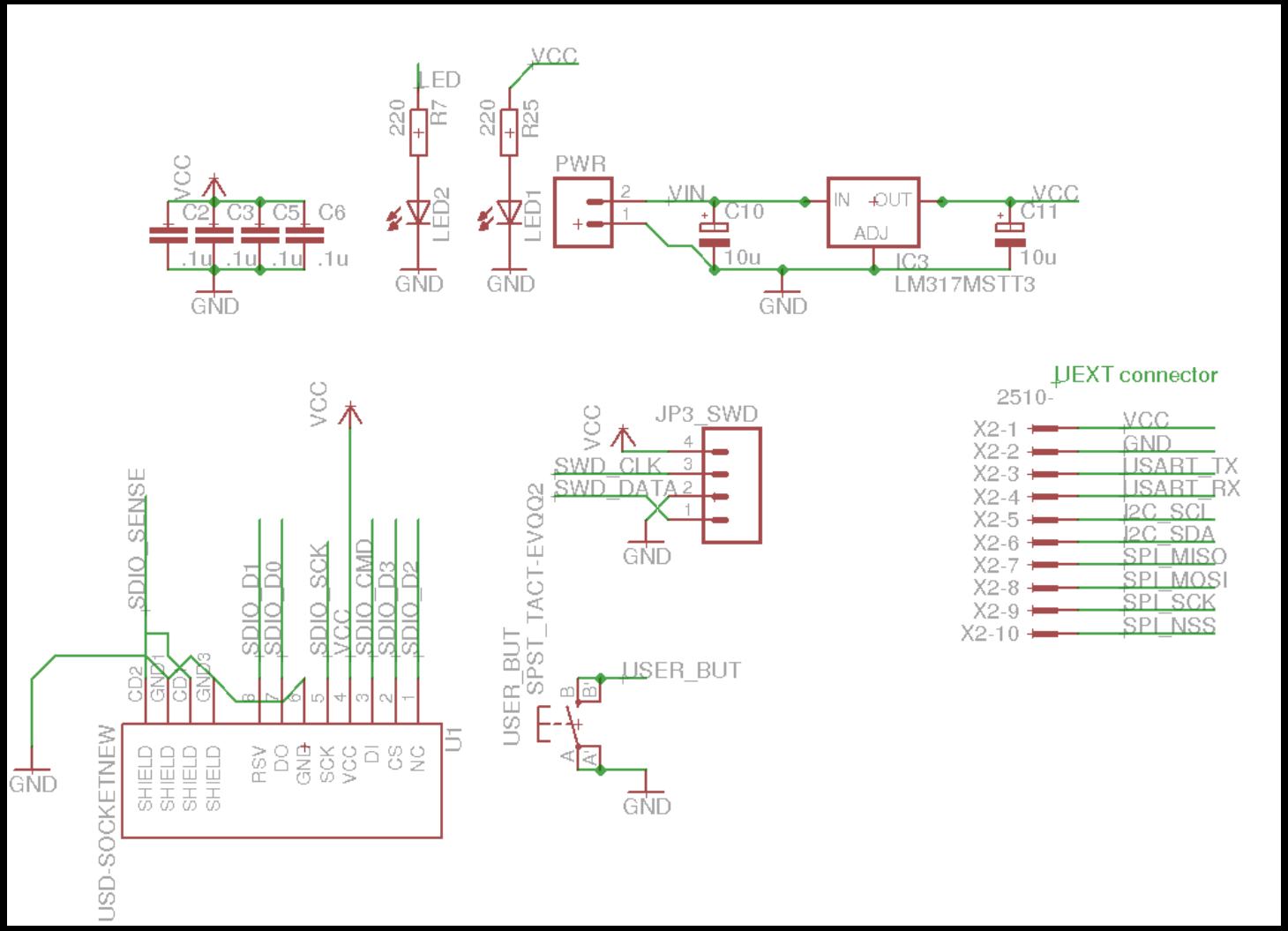
- internal dual DAC STM32F4 can be used as is !
  - Tsettling : typ 3µs (300kHz)
  - write 2xu8 (or u12) → output !
  - Max 12 bits output

- Simple passband filters to remove noise and filter DC
  - Stereo : double circuit
  - No amp : direct drive MCU

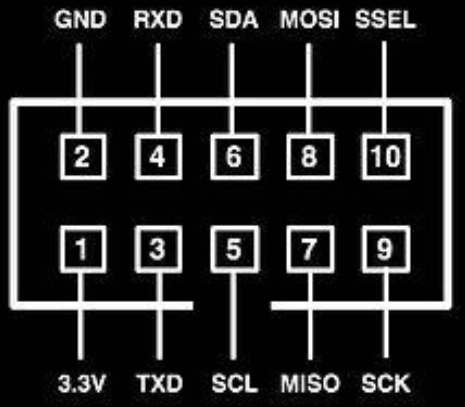
Sound, USB


VGA_R
VGA_G
VGA_B
X1-1
X1-2
X1-3
X1-4
X1-5
X1-6
X1-7
X1-8
X1-9
X1-10
X1-11
X1-12
X1-13
X1-14
X1-15
VGA_H
VGA_V
GND
SPST_TACT-EVQQ2
RESET
C1
.1u
GND
USBAB
OTG_FS_ID
OTG_FS_DP
OTG_FS_DM
ID
D+
D-
VBUS
FU01
500mA
VIN
U$2
D+1
D-1
VBUS1
GND1
D+2
D-2
VBUS2
GND2
OTG_HS_FS_DP
OTG_HS_FS_DM
STACKED_USB
C7
22u
C8
1u
R26
536r
R6
1070r
AUDIO_L
U2
C4
C14
R15
R21
AUDIO_R

Power , SDIO, expansion port, LEDs



Expansion port



Hardware Hacking : DIY on DIY !

Exposes ports / interfaces of the MCU

Shield arduino : Big, not very useful

UEXT : existing “standard”

- UART, SPI, I2C, Vcc (GPIO)
- Simple connector (2.54, 5*2)
- Existing : IrDa, RPI, LCD, Bluetooth, GSM, GPS, Eth + POE, Wifi, biofeedback, Gyro, Thermo, Wii nunchuk ...
- DIY ready ! - https://en.wikipedia.org/wiki/UEXT

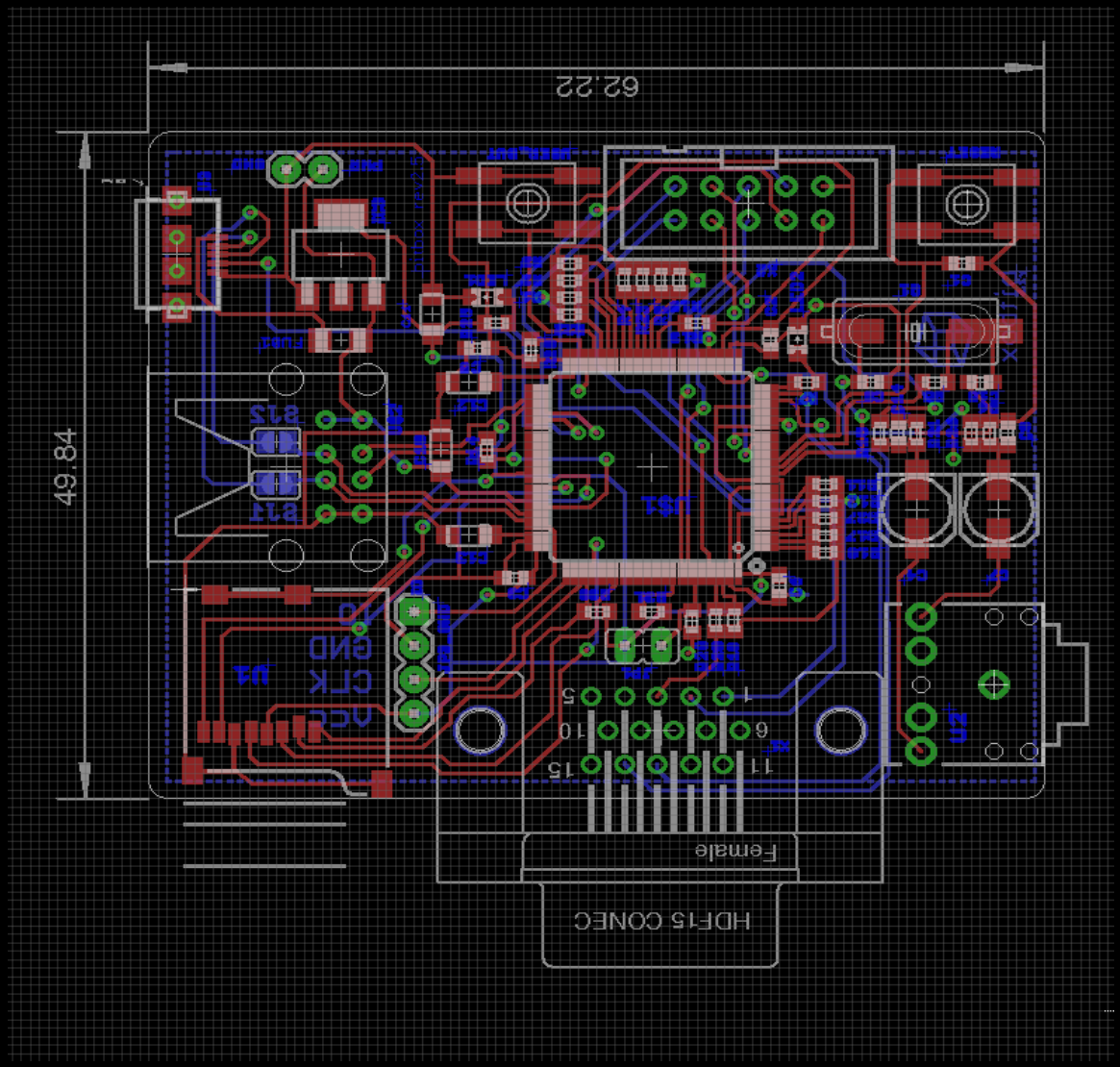
STM32F40x

STACKED_USB

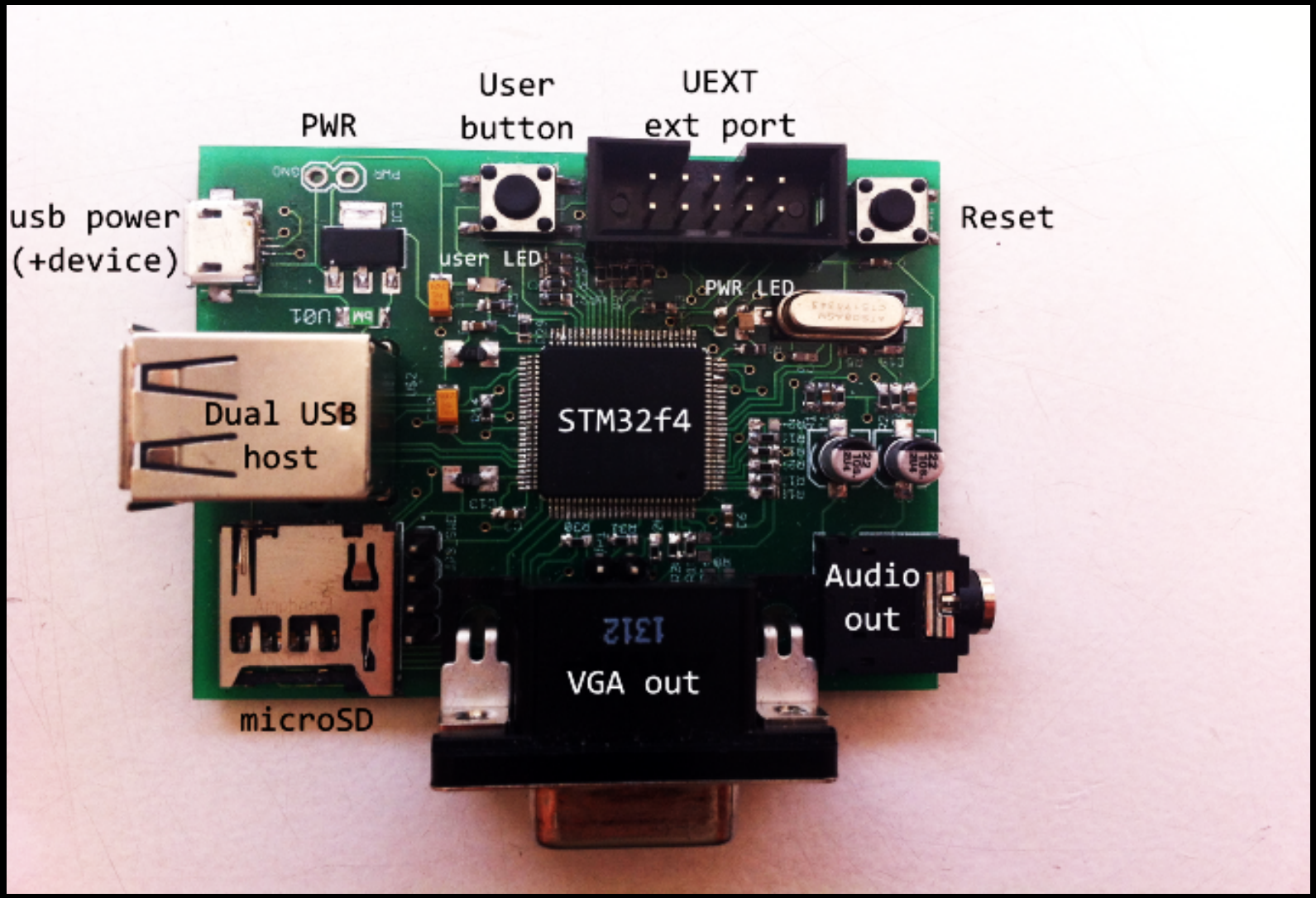
USB-SOCKETNEW1

changed the adio pinos
todo xhange orientation sd
change orientation sound ?
pladler & reinverser boot pins

Bitbox Hardware



Bitbox Hardware



2 - Software

toolchain : arm-none-eabi-xxx

- gcc : .c → .o

-mthumb -mcpu=cortex-m4 -mfloat-abi=hard -mfpu=fpv4-sp-d16 -march=armv7e-m

- linker (ld) : *.o → .elf
- objdump : generates binaries
- gdb : on-chip debugging !
- ubuntu ✔, windows ✔, mac : ?

the full Software Stack

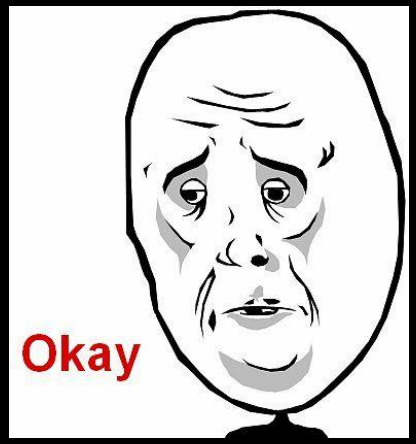
the full Software Stack

YOUR GAME*

*(with VGA setup, hardware register setup, clock tree init, sound DMA setups, USB stack, HID parser, SDIO debug)

(not fun for everyone)

the hardware.



the full(er) Software Stack

Kernel

- SDK / API
  - Small set of Low level drivers = a set of optional libraries
  - Lib compiled in game - bitbox.h
  - No dependency, Open Source
  - No specific algorithms (blitter, engine, mixer ...)
- Video
  - init hardware (DMA, timers, clocks),
  - send data to DMA & drive DMA with timers
  - Interface “blit_next_line(u16 *)”
- Audio
  - Initialize & drive : DMA / timers / DAC
  - Interface : fill_buffer(u8*)
- SD
  - includes Fat32 lib (fatfs) + SDIO HW interface
- USB
  - simple interface to hide complexity
  - HID, drivers Keyboard/Mouse/Gamepads

Kernel : emulator

SDL Implementation of bitbox.h interface

Source-compatible, same API

Gamepads, Video, Audio, Keyboard, SD ...

What compiles and runs with bitbox.h can be emulated as is

(need to be careful with reciprocal ...)

Kernel : emulator

SDL Implementation of bitbox.h interface

Source-compatible, same API

Gamepads, Video, Audio, Keyboard, SD ...

What compiles and runs with bitbox.h can be emulated as is

(need to be careful with reciprocal ...)

(but a game has been made just with it)

(and it worked first time on the bitbox)

the (emulated) more full(er) Software Stack

graphics compression

- Necessary
  - 640x480*16bits : 614 ko ! almost whole flash
  - 1 DMA = 1 line, not one screen
  - On the fly decompress to line buffers

- Needs to be quick
  - 7 cycles/pixels
  - cannot keep result from one frame in memory
  - MPEG has interfame compression ...

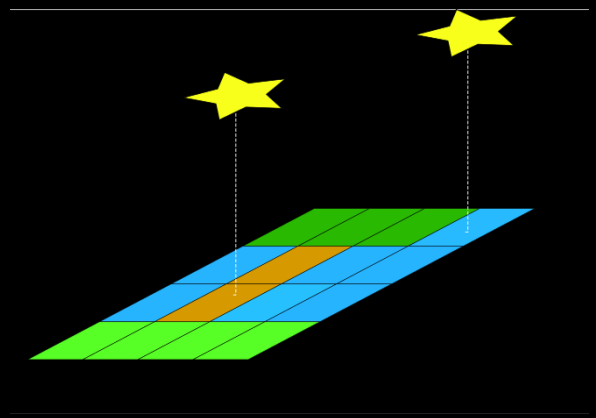
- Sometimes simple to modify (GFX should be modified by game engine)
  - counter ex : RLE lines, zlib, LZ4
  - not always

engines

- different uses, different solutions
  - engine avoid duplicating
  - relies on top of kernel repose (emulable !)
- Text mode
  - 80x25 bytes buffer characters (VGA text) or 132x75
  - bitmap font in RAM
- 64kB Frame Buffer
  - reduces simultaneous colors or resolution
  - simplest to use (“write to memory”)
  - 800x600 2c (1bpp) → … → 320x200 8bpp
- 2D engine : scene composition: Tiling + sprites (next slide)
- Procedural ? Vector ? Video ? : possible
- game specific, mix n match : software and open !

2D Engine

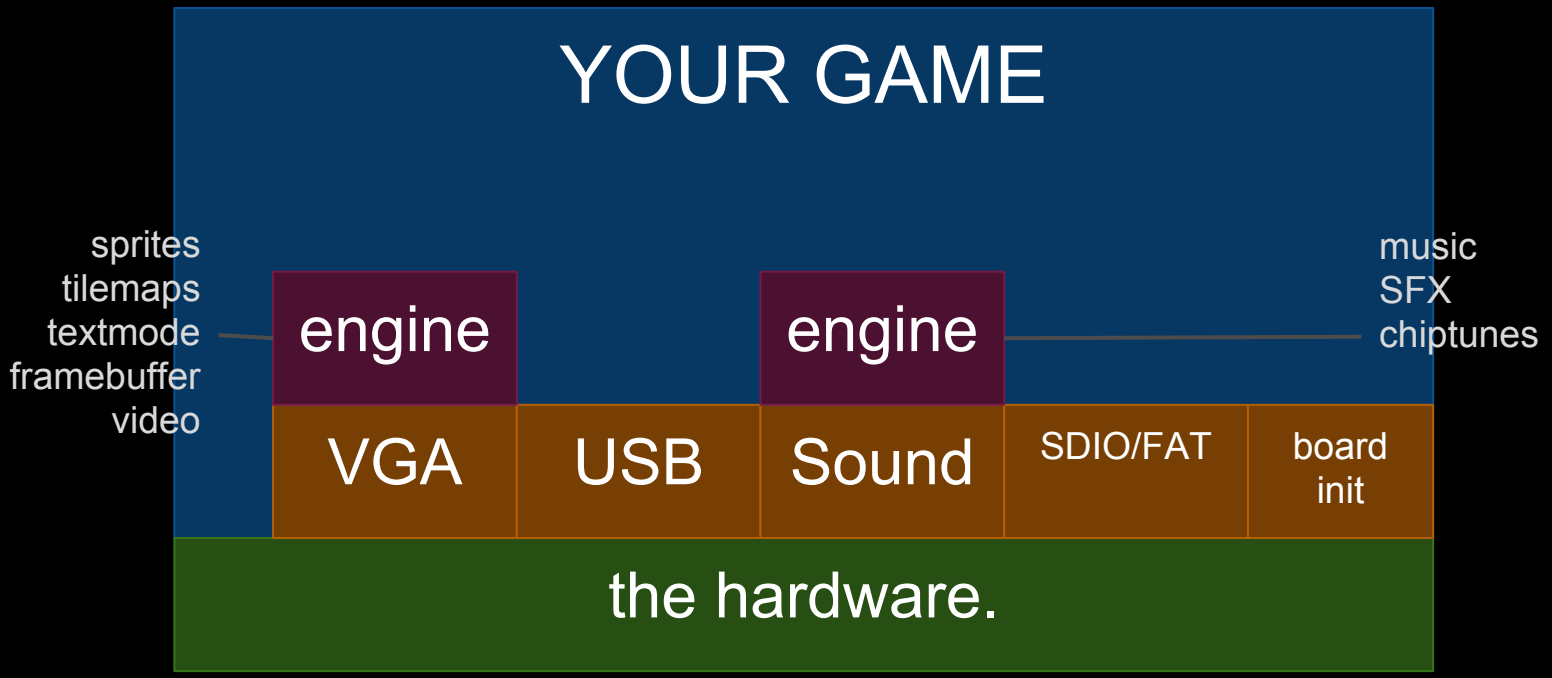
- composing a simple scene with a simple API
- different objects with render line func (in C)
  - rectangles
  - sprites (u16, c8, p4, rle)
  - tilemaps (RAM and Flash)
- Stored in RAM or Flash
- Blit top down (z order)
  - keeping track of blit progress



sound engines

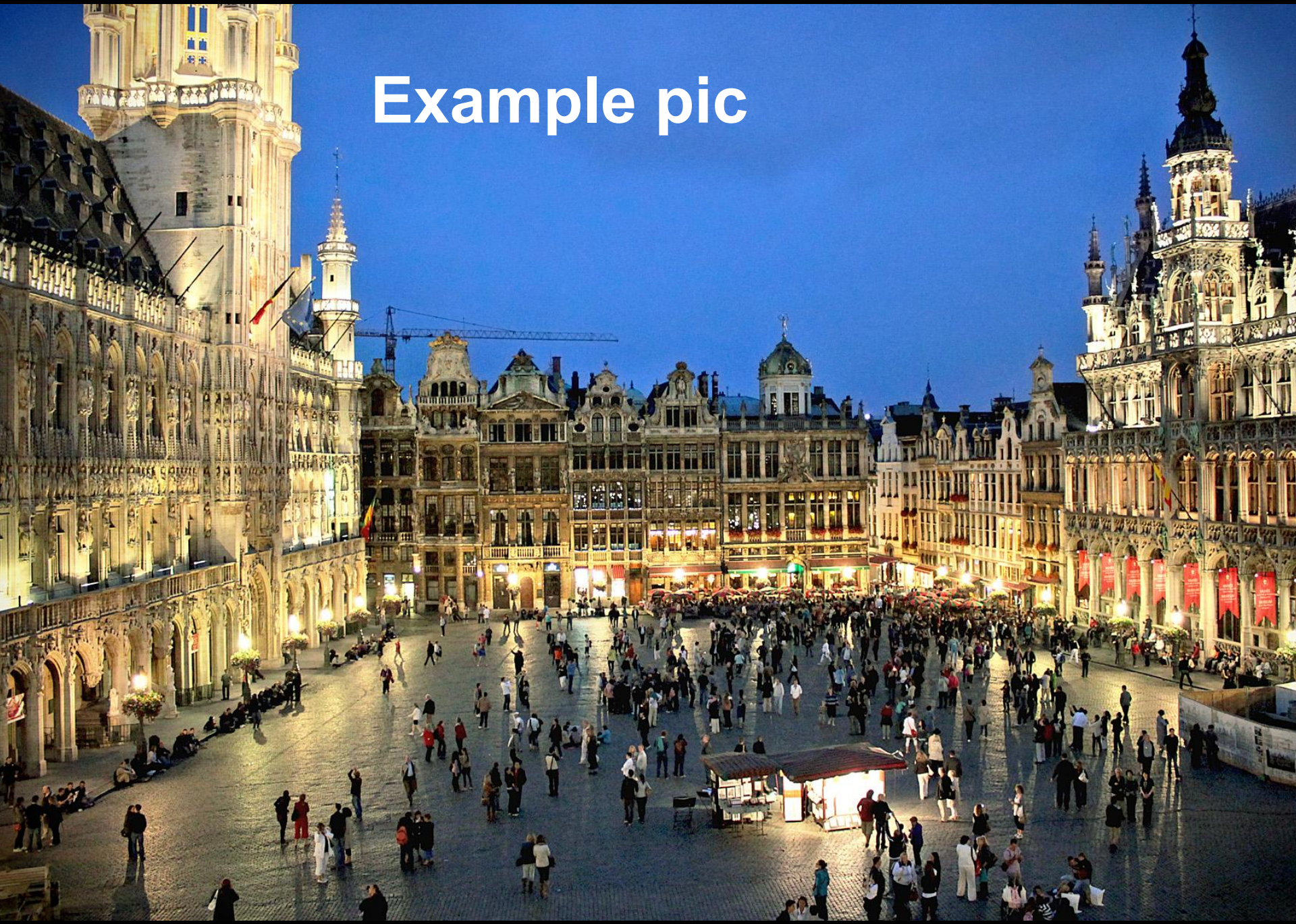
- Simple Sampler
  - Play & resample audio from flash (notes/SFX)
  - midi file imports + triggers
  - stream sounds from uSD card
- MOD player !
- Simple Synths (tri, square)
  - with tracker
- ...

the (more) full(er) Software Stack



full motion video

- streaming uSD → buffer → screen lines
- 1 frame stored compressed in RAM !
  - no interframe compression
  - line decompression
  - 7 cycles per pixel
- develop a new codec : **btc4**
  - 256c palette
  - 4x4 squares (16 pixels) : 2 colors/square (1bpp)
  - 2 index u8 + 16 bits = 2bpp !
  - 640x360 = 57kB : in RAM !
  - python encoder, speed and quality can be improved
- Stream compression : reduces SD throughput


Example pic

Video example

https://www.youtube.com/watch?v=39QEnIZX

program transfer

- Internal Bootloader (ROM)
  - transfer via USB device or serial → flash
  - autonomous, usable on bitbox

- SWD
  - Debug interface
    - direct memory access
    - breakpoints hardware
  - swd port exposed on bitbox
  - STM32 Discovery card (<10€)
  - linux : st-util + gdb remote, st-flash

- **SD card ?**

bootloader

bootloader : resident, small, stable, safe

select file on screen with gamepad & loads from uSD to flash

bootloader functions:

FAT, SDIO, USB, VGA, Flash programming, text output ...

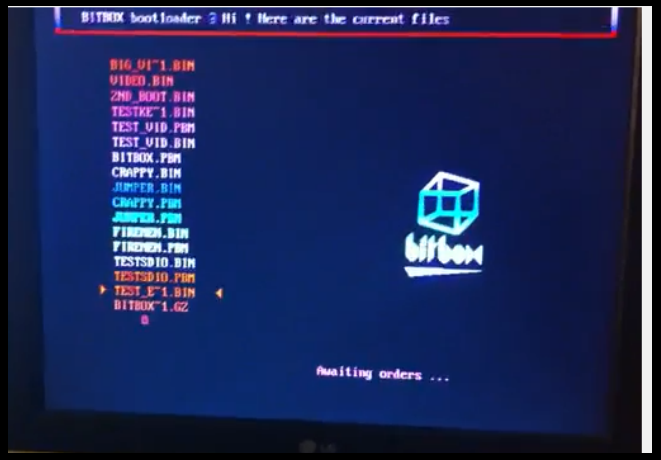
⇒ complex (Heavy, Updated)

⇒ cannot run from flash !

2-stage bootloader

1. bootloader1 : loads second to RAM
   - only SD/FAT minimal
   - always on flash, <16k

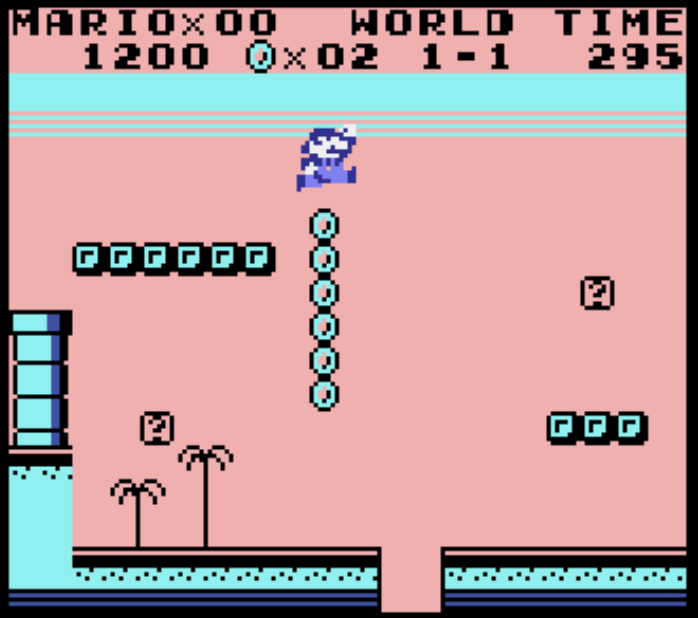
2. bootloader2 :
   - runs from RAM
   - complex, can be updated
   - load program to flash
   - uses standard bitbox libs



programs

BitBoy

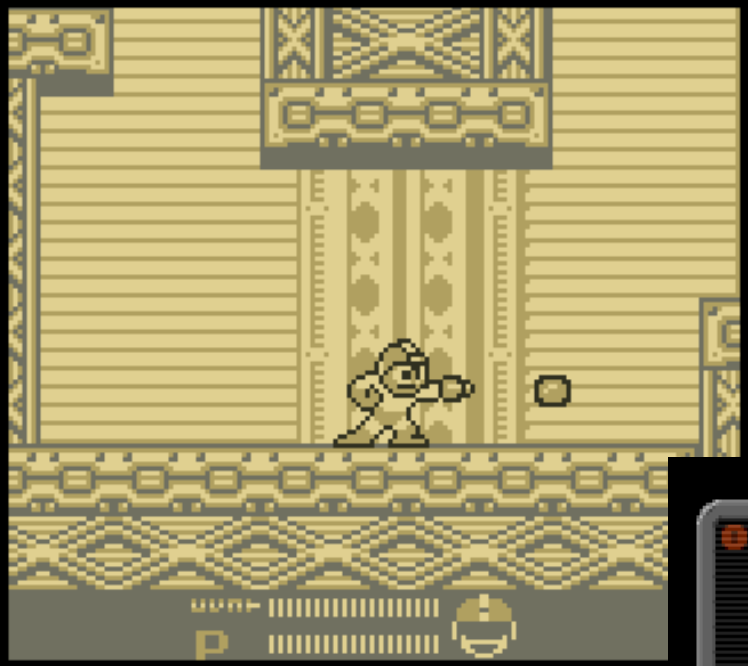
- Port of Gnuboy, a Gameboy emulator
  - smallest gnuboy port
  - redone blitter engine
  - Axing functionnality
  - Games in flash (not loaded from SD)



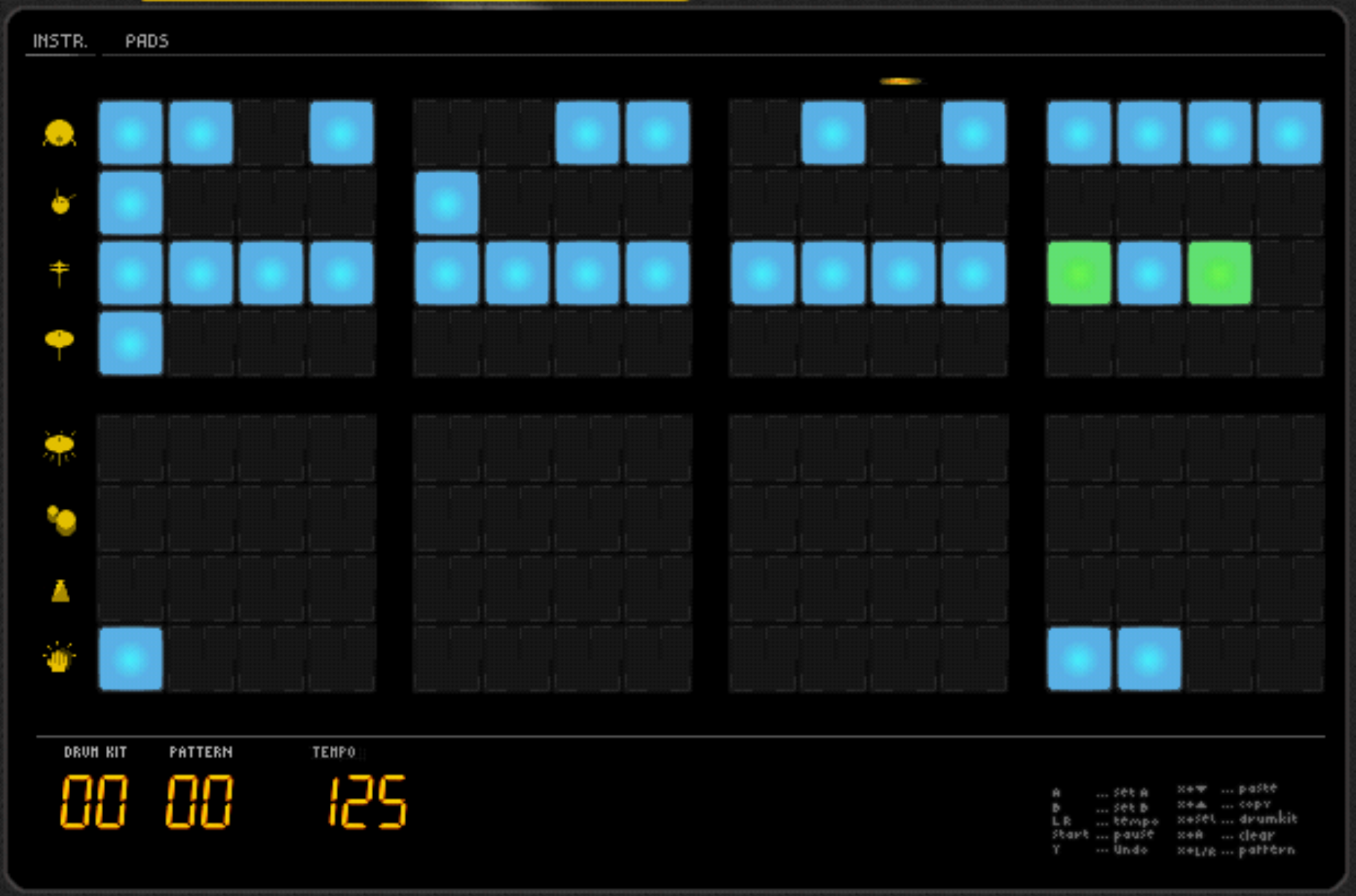
- mario, metroid, … work !
  - also on emulator

- todo :
  - RAM games with SD paging
  - color gameboy
  - NES !

Games screenshots

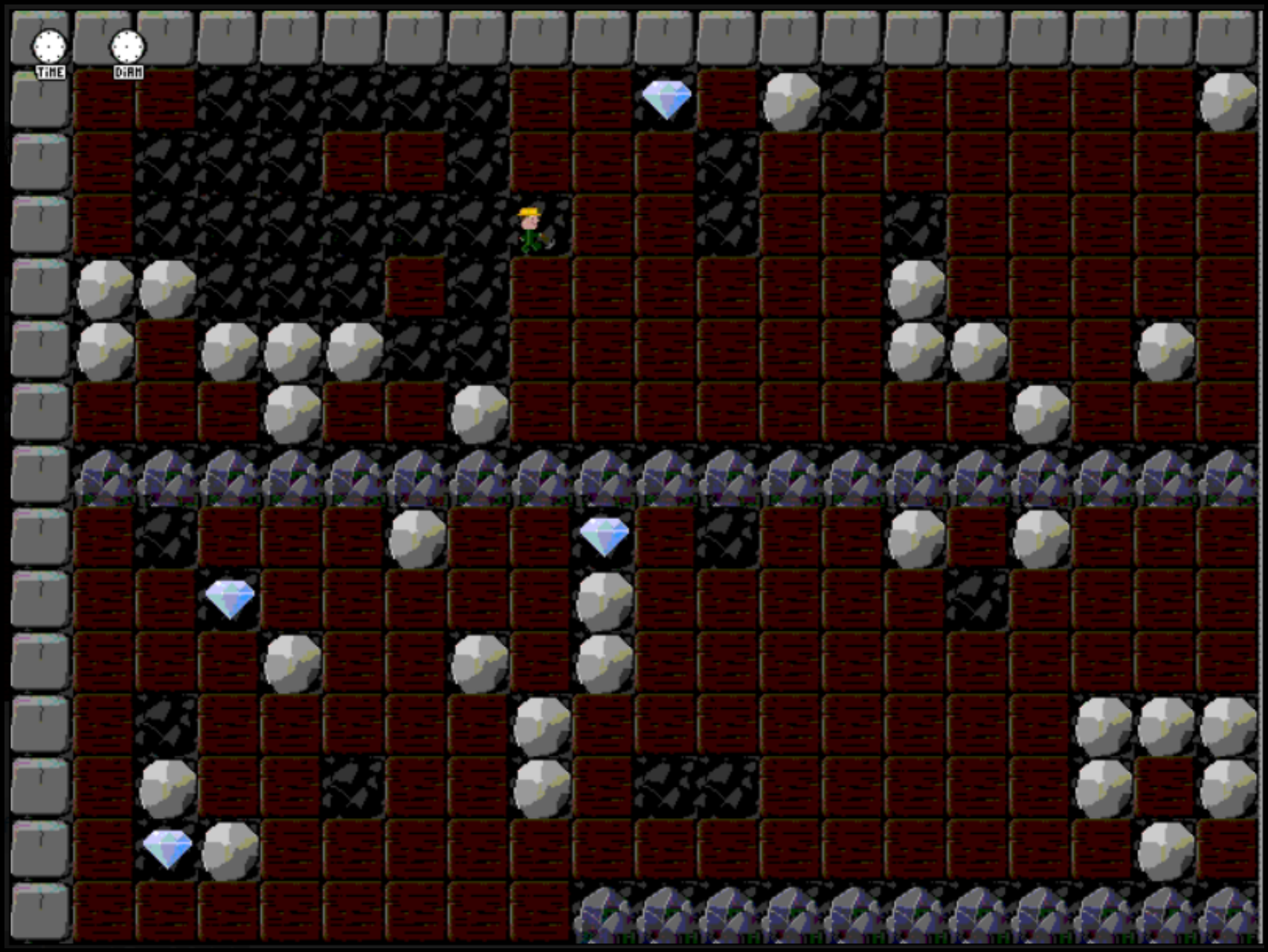


Beat Blocks


INSTR.
PADS
DRUM KIT
PATTERN
TEMPO
00
00
125
A ... set A
B ... set B
LR ... tempo
Start ... pause
Y ... Undo
×+▼ ... paste
×+▲ ... copy
×+Sel ... drumkit
×+A ... clear
×+LR ... pattern


160


NYAN
! !
WORLD
RECORD
0334
WEEE! !


TIME
DIAM

other games / softs

- 2048, Crappy bird !
- Mod player, Video player, SDK examples, USB devices tester ...

- In progress : *many* (too much) !

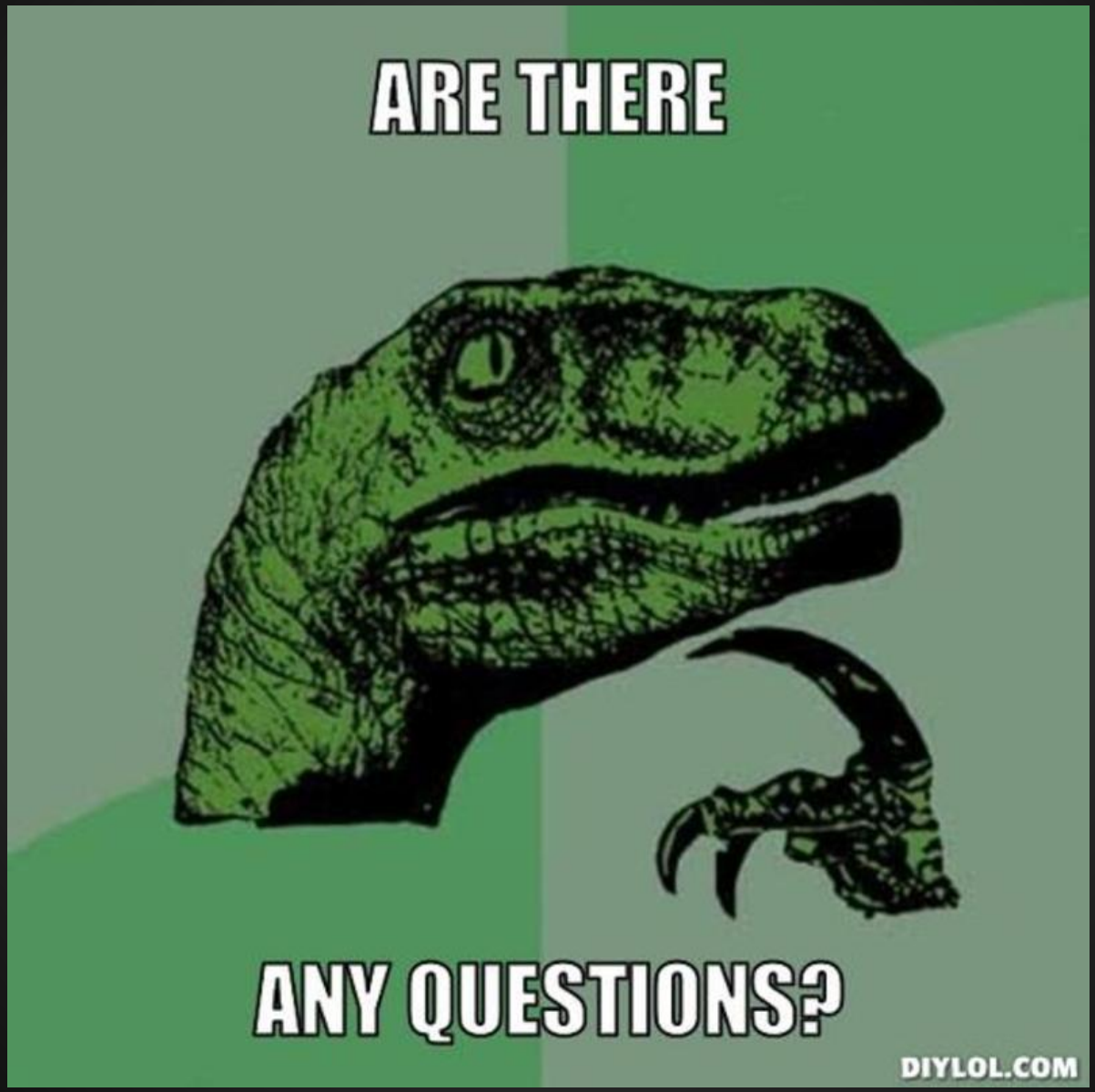
⇒ Shmup, Video adventure, Beam Racer, Bomberman, Shadow of the Bit…

future ?

- Games ! (Emulator / Engine)
  - gfx, sounds, ports ...
- Debug …
- Improve kernel / engines
- 8bits emulators ! (Nes, ZX, …)
- Extensions
  - Wifi (ESP8266)
  - Nunchucks
- not more powerful hardware (stay compatible)

links

bitboxconsole.blogspot.com
https://github.com/makapuf/bitbox


ARE THERE
ANY QUESTIONS?
DIYLOL.COM