

# Arquitecturas y Entornos de desarrollo para Videoconsolas

Grado de Ingeniero en Informática  
Escola Tècnica Superior d'Enginyeria Informàtica  
Curso 2015/2016

# Objetivos

- Conocer la arquitectura hardware de la plataforma NDS
- Identificar las rutas de datos e instrucciones
- Reconocer el uso de los componentes de la arquitectura de los que se hace uso en las aplicaciones existentes
- Aplicar las estrategias existentes a los propios desarrollos
- Conocer y saber dimensionar una aplicación para esta plataforma

# Índice

- Introducción
  - Características hardware de la familia NDS
  - Ejecución de aplicaciones propias
- Arquitectura de la plataforma
  - Arquitectura hardware de NDS
  - Arranque de la NDS
  - Arquitectura software: DevkitPro + *libnds*
- Estructura de una aplicación
  - Estructura mínima
  - Un ejemplo complejo (*DSOrganize*)

# Características de la NDS

- Game Boy
  - 8-bit Sharp LR35902 (compat. Z80), 4,19MHz
- Game Boy Advance (GBA)
  - ARM7TDMI, de 32 bits + Z80, soporte a la GameBoy clásica
- Nintendo DS (NDS)
  - Z80 → ARM9 a 33Mhz +  
↑↑2D y motor 3D
- GamePark 32 (GP32)
  - ARM920T + Tarjetas SD.



# Características de la NDS

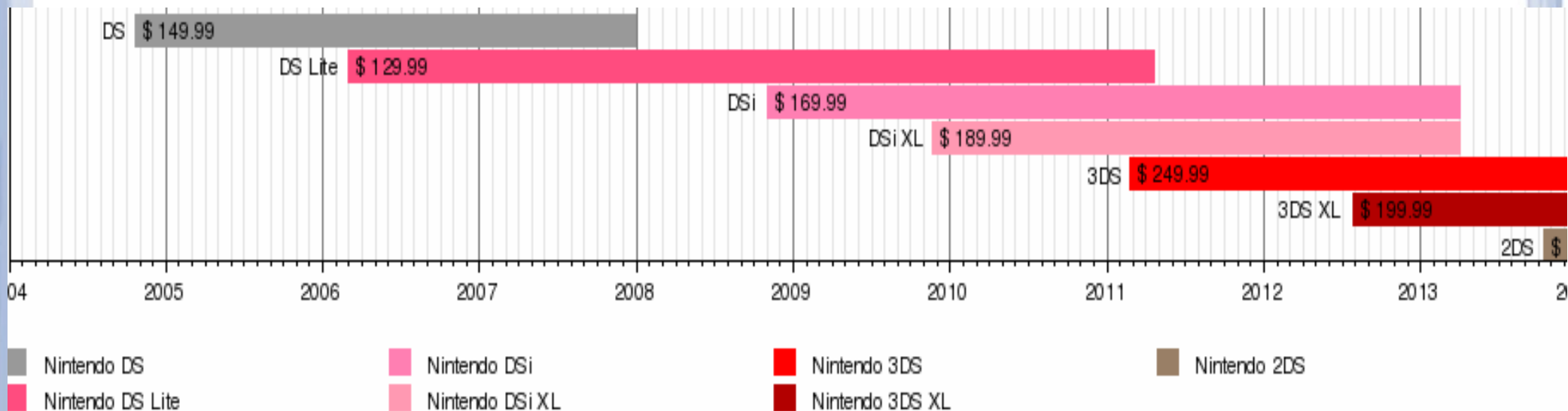
- *NDS / NDS Lite*
  - “*Lite*” y 4 niveles de brillo
- *NDSi / NDSi XL*
  - 2 cámaras
  - ↑↑prestaciones , tarjetas SD, ↓Slot2.
- *3DS / 3DS XL*
  - Pantalla 3D
  - Acelerómetro, giróscopo, IR
- *2DS*
- *VC*

*Virtual Console*™



# Características de la NDS

- Disponibilidad actual
  - No se fabrican DS/DS Lite



- Comprobar cambios en la arquitectura para fijar objetivos



# Características de la NDS

- *NDS*

- 256x192px, 262,144 ( $2^{18}$ ?) colores,  
2 niveles brillo, 67 MHz ARM946E-S  
+ 33 MHz ARM7TDMI, 4MB, 802.11 (*legacy mode*)



- *NDS Lite*

- 4 niveles brillo

- NDSi / NDSi XL

- 133 MHz ARM9 + 33 MHz ARM7, 16MB,  
802.11b/g, 2x0.3 Mpx cámaras

- 3DS / 3DS XL

- 800x240px + 320x 240px, 16,7Mcolores, 5 niveles brillo,  
dual-core ARM11, GPU PICA200128MB, 3x0,3Mpx  
cámaras, acelerómetro, giróscopo, IR



# Ejecución de aplicaciones propias

- *Desarrollo de aplicaciones para esta plataforma*
  - *SDK “oficial”*
    - “Software Development Suport Group” (SDSG)
    - *Gestiona el acceso a la documentación y SDK.*





# Ejecución de aplicaciones propias

- *Desarrollo de aplicaciones para esta plataforma*
  - *SDK “oficial”*
    - “Software Development Suport Group” (SDSG)
    - “To become an Authorized Developer for Nintendo game platforms”.

Imágenes obtenidas del sitio web de SDSG:

<<http://www.warioworld.com/>> ,



- “To apply for Wii U, Nintendo 3DS, Wii/WiiWare, or Nintendo DSi/DSiWare, you will need to complete an updated Developer Application for your company.”

– *Firmware de la NDS*

# Ejecución de aplicaciones propias

- ¿Novedades en el SDK oficial?

- Nintendo Developer Portal <<https://developer.nintendo.com/>>

personajes-de-nintendo-280983/

- ¿Nintendo Developer Program?

- ¿Ecosystem of specialized tools?

- ¿Nintendo Dev Interface (NDI Client)?

- ¿NX?

- Register for access to Nintendo developer tools, resources, downloads and more, all to help you create and publish Nintendo games and applications using HTML5, Unity, or native Wii U and 3DS SDKs.

Enroll in the Nintendo Developer Program



- For individuals, Nintendo offers the Wii U Developers Program. This program is focused on development of games for the Nintendo eShop on Wii U using Unity or HTML5. ... No prior development experience is required.

# Ejecución de aplicaciones propias

- *Desarrollo de aplicaciones para esta plataforma*
  - *Middleware* <<https://developer.nintendo.com/game-developers>>
    - *Middleware can simplify and aid your game development, providing services as diverse as game engines, compressions tools, video codecs, sound players, and more. In addition to a large array of 3rd-Party middleware that's available for our systems, Nintendo has gone the extra step and licensed some middleware to our developers for free or reduced cost. Want to use Unity for Wii U to build your Wii U game? You can do that. With the Nintendo Web Framework, you can build your game or app in HTML 5, as well. From Havok Wii U XS to PUX for both Wii U and Nintendo 3DS, we've got quite the selection to meet your needs.*
    - *Havox* <<http://www.havok.com/>>
      - *is an Irish computer software company that provides interactive software and services for digital media creators in the video game and movie industries: physics, destruction, cloth, AI, ...*
        - *Microsoft (2015) ← Intel (2007)*
    - *PUX* <<http://pux.co.jp/en/casestudy/>> (Panasonic, Nintendo 27% en 2013)
      - *Codec (3GPP / 3GPP2)*
      - *LiteSpeech / LiteSpeech Advance (SR / TTS) → brainTraining; OCR*
      - *SoftSensor Image Recognition Software*
        - *Handwriting, face recognition, scene recognition, object (hand / finger) recognition, gesture recognition*

# Ejecución de aplicaciones propias

- *Desarrollo de aplicaciones para esta plataforma*

- *“Become a Wii U Developer!”*

- *<<https://wiiu-developers.nintendo.com/>>*

- *Unity for Wii U*

- *Nintendo pays for the Unity License on Wii U for all licensed Nintendo Developers.*

- *Nintendo Web Framework*

- *Founded on WebKit technologies and harnessing common programs — including HTML5, JavaScript, and CSS — it allows development to span across the Wii U GamePad, Wii Remote controllers, and more.*



Maximize your HTML5 development by using middleware!



Construct 2



# Ejecución de aplicaciones propias

- *Firmware de la NDS*
  - Nintendo's own firmware boots the system.
    - A health and safety warning is displayed first, then the main menu is loaded.
    - The main menu presents the player with four main options to select: play a DS game, use PictoChat, initiate DS Download Play, or play a Game Boy Advance game.
      - The main menu also has some secondary options such as date and time, GBA screen, and touchscreen calibration.
      - Also features an alarm clock, several options for customization ..., and the ability to input user information and preferences (such as name, birthday, favorite color, etc.) that can be used in games.
- *FlashCards / FlashCarts*



# FlahsCards / FlashCarts

- *Firmware*
  - *Cargar aplicaciones desde el FlashCard*
    - *Ejecución de juegos oficiales*
    - *Copias de seguridad*
    - *Ejecución de aplicaciones no oficiales*
      - *SDK no oficial o “homebrew”*
  - *Simon van de Berg. 2006. Running Nintendo DS homebrew*
    - *Pirating of software is something I do not approve of.*
    - *Pirating is often associated with homebrew. Pirating is a term used for running official games you do not own, or do own, but are not allowed to play in some way by law.*
    - *Homebrew is creating and sharing programs made by yourself and/or others for free. This means that no business is attached to the software. ... no support ...*
- *Tipos: DS Slot vs GBA Slot*



# FlashCards / FlashCarts (y II)

- *Flash cartridges*

- Dispositivos de almacenamiento NDS

- *32 MiB block of rewritable flash memory directly-accessible by both CPUs of the Nintendo DS.*

- Tipos

- En función del puerto o *slot*

- Slot-1: DS Slot
    - Slot-2: GBA Slot
      - Desaparece en DSi



# Arquitectura de la NDS

- Hardware

- Componentes

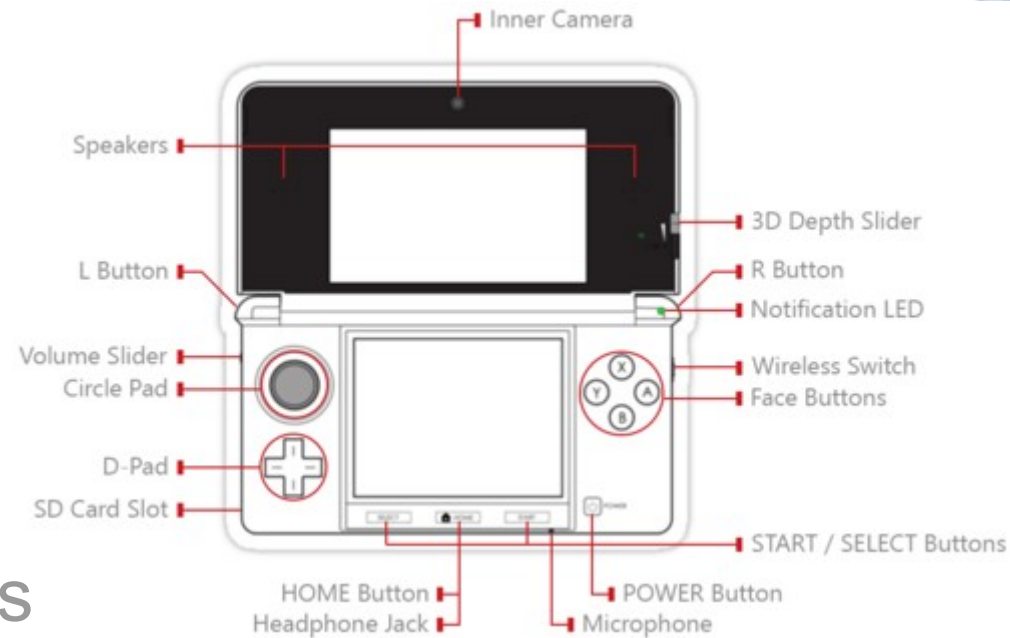
- Compatibilidad hacia atrás
    - Tendencias



- Esquema de bloques

- Software

- SDK



# Componentes

- Vista trasera

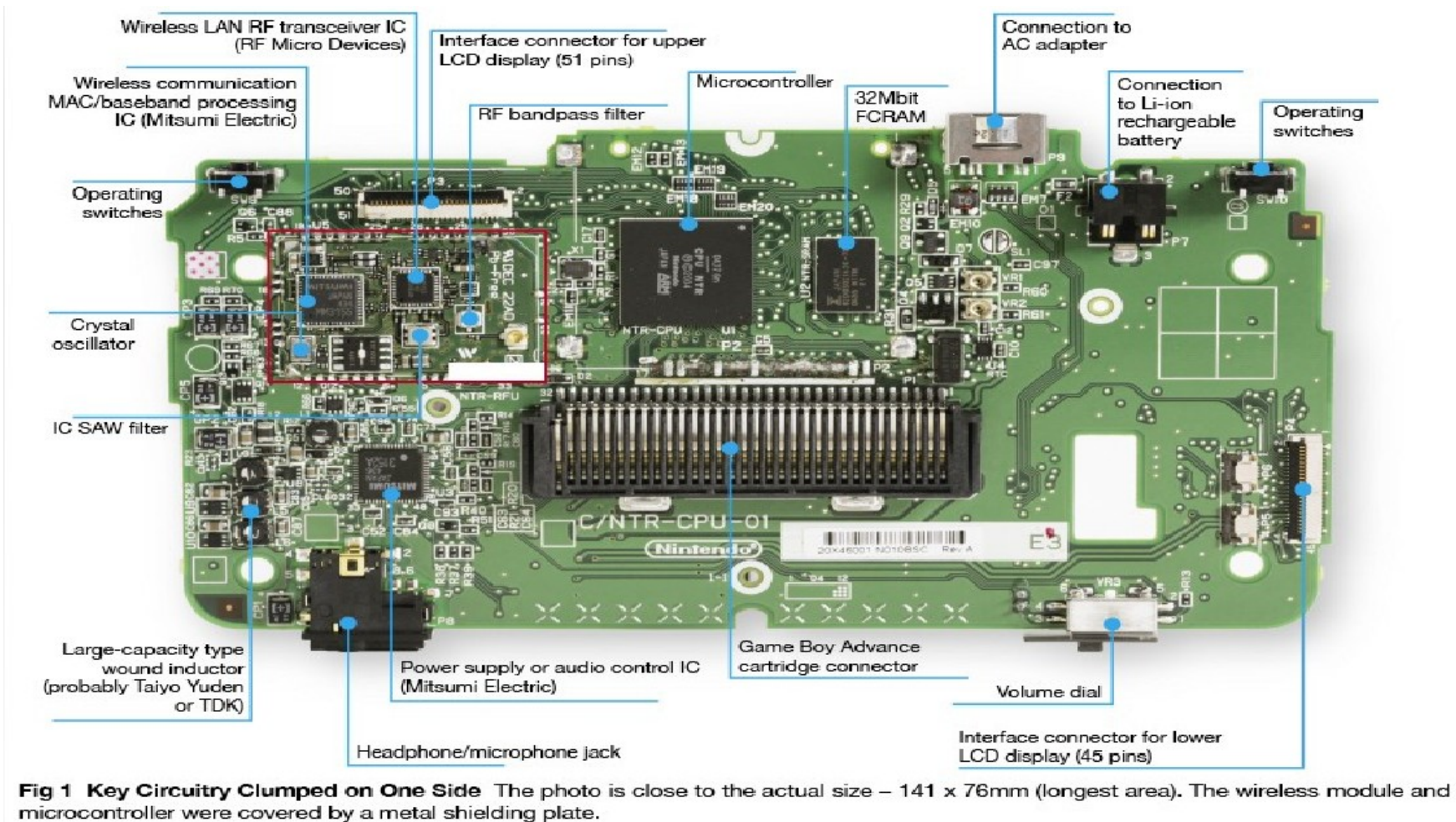


Imagen obtenida de

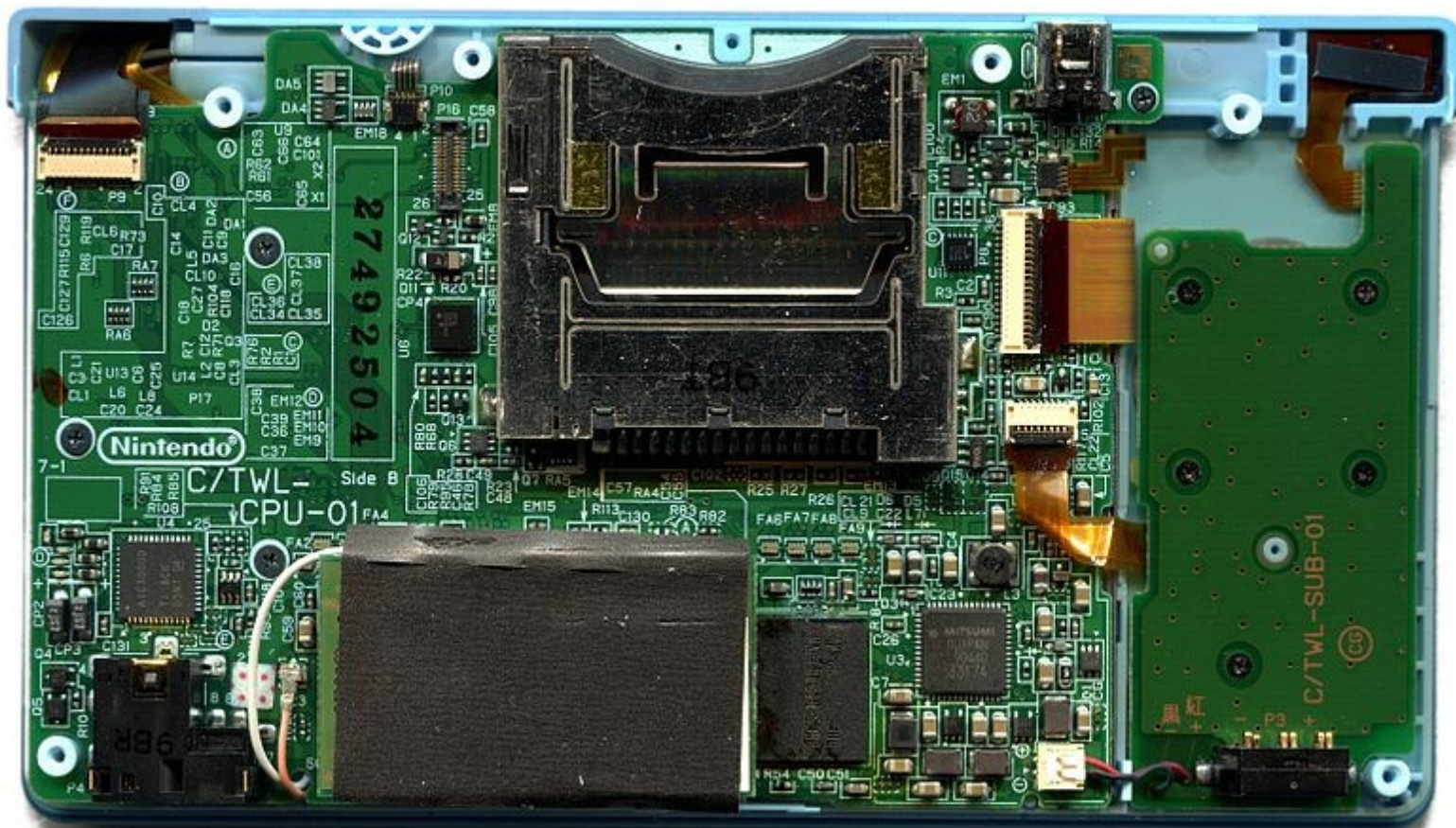
"CS4803 Design and Programming of Game Consoles, Spring 2011" <<http://www.cc.gatech.edu/~hyesoon/spr11/index.html>>

<<http://techon.nikkeibp.co.jp>>



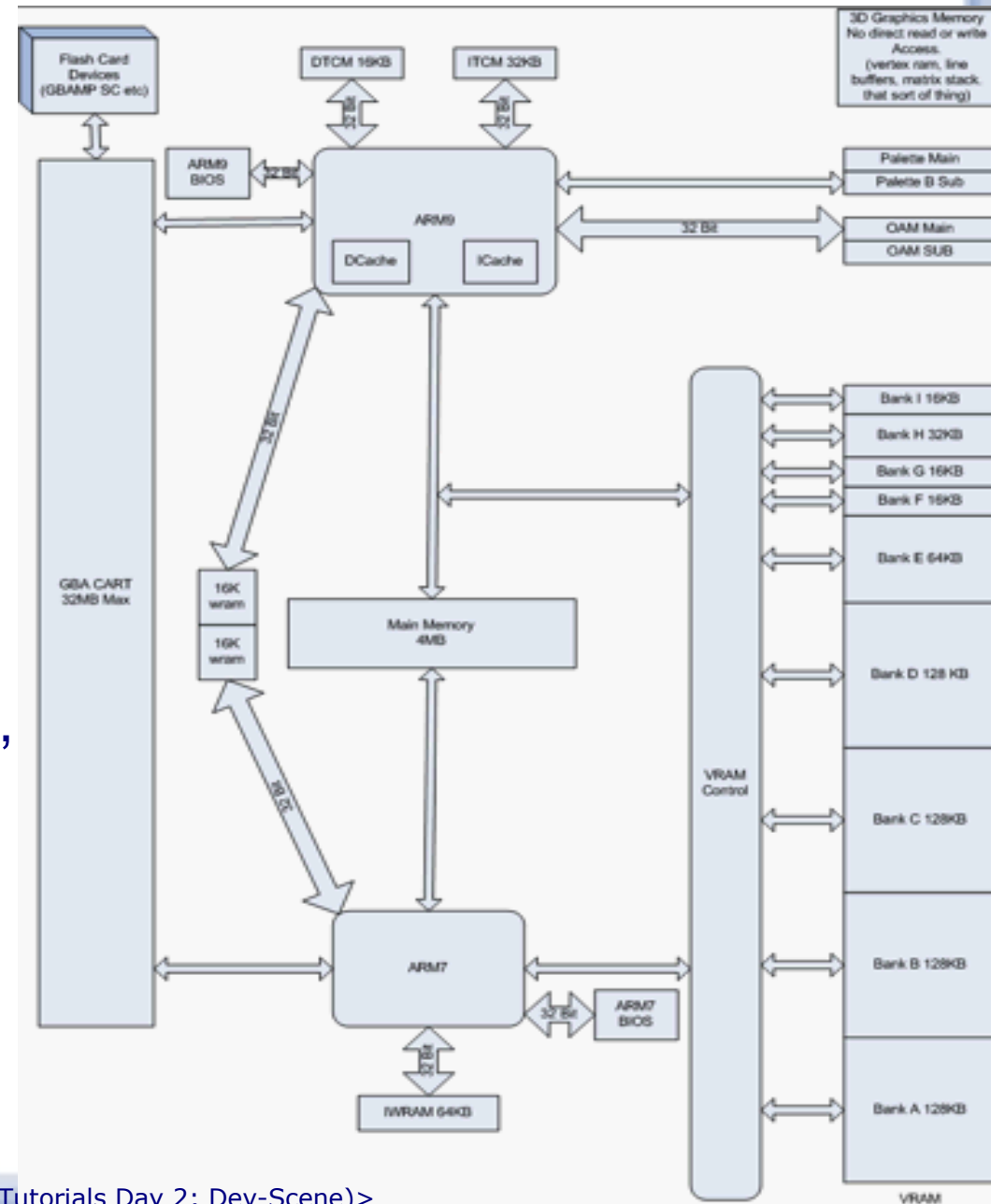
# Componentes (y II)

- Vista frontal (NDSi)



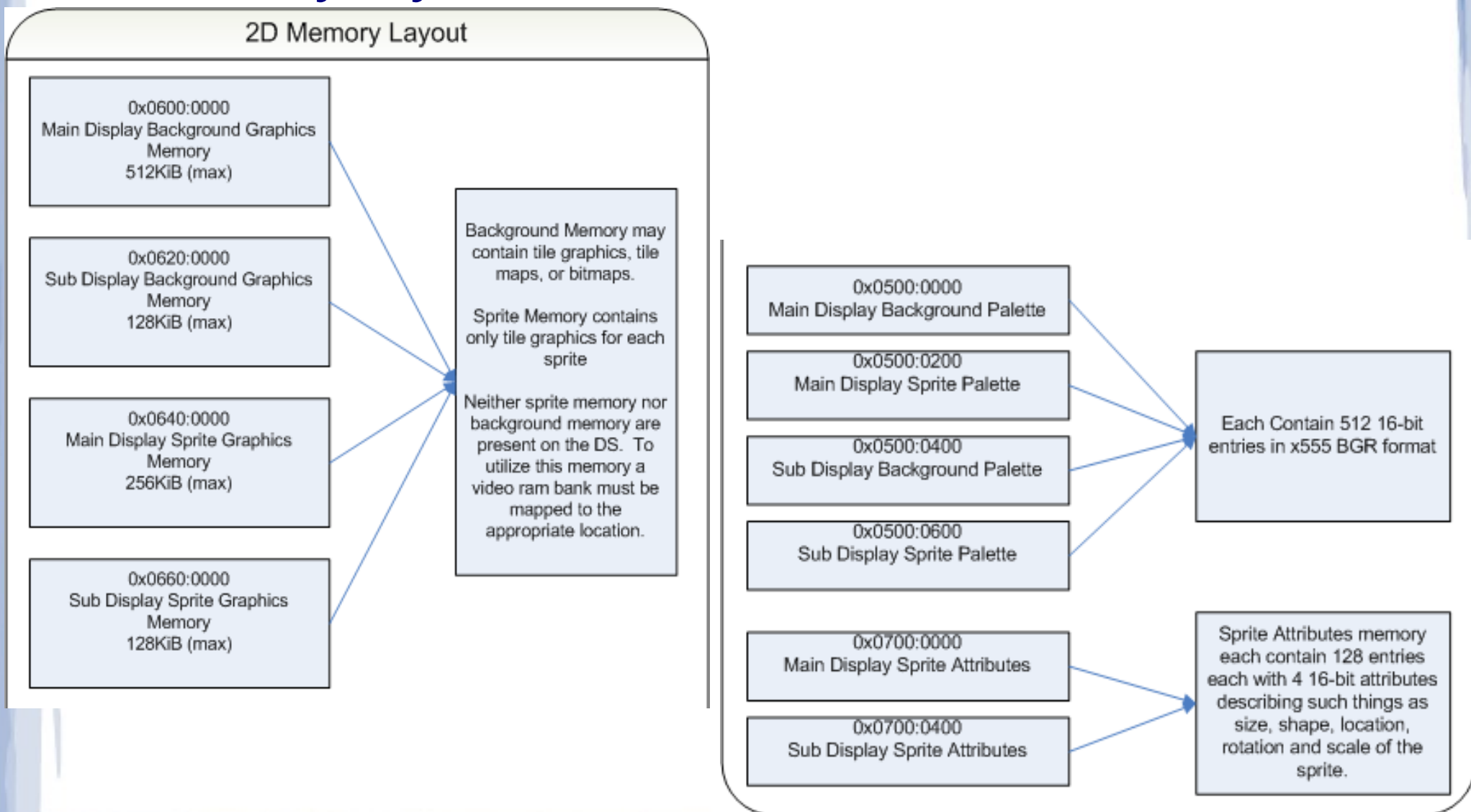
# Esquema de bloques

- Conexión entre elementos
  - Procesadores
  - Gestión de memoria
  - Subsistemas
    - Audio
    - “Aceleradores” gráficos



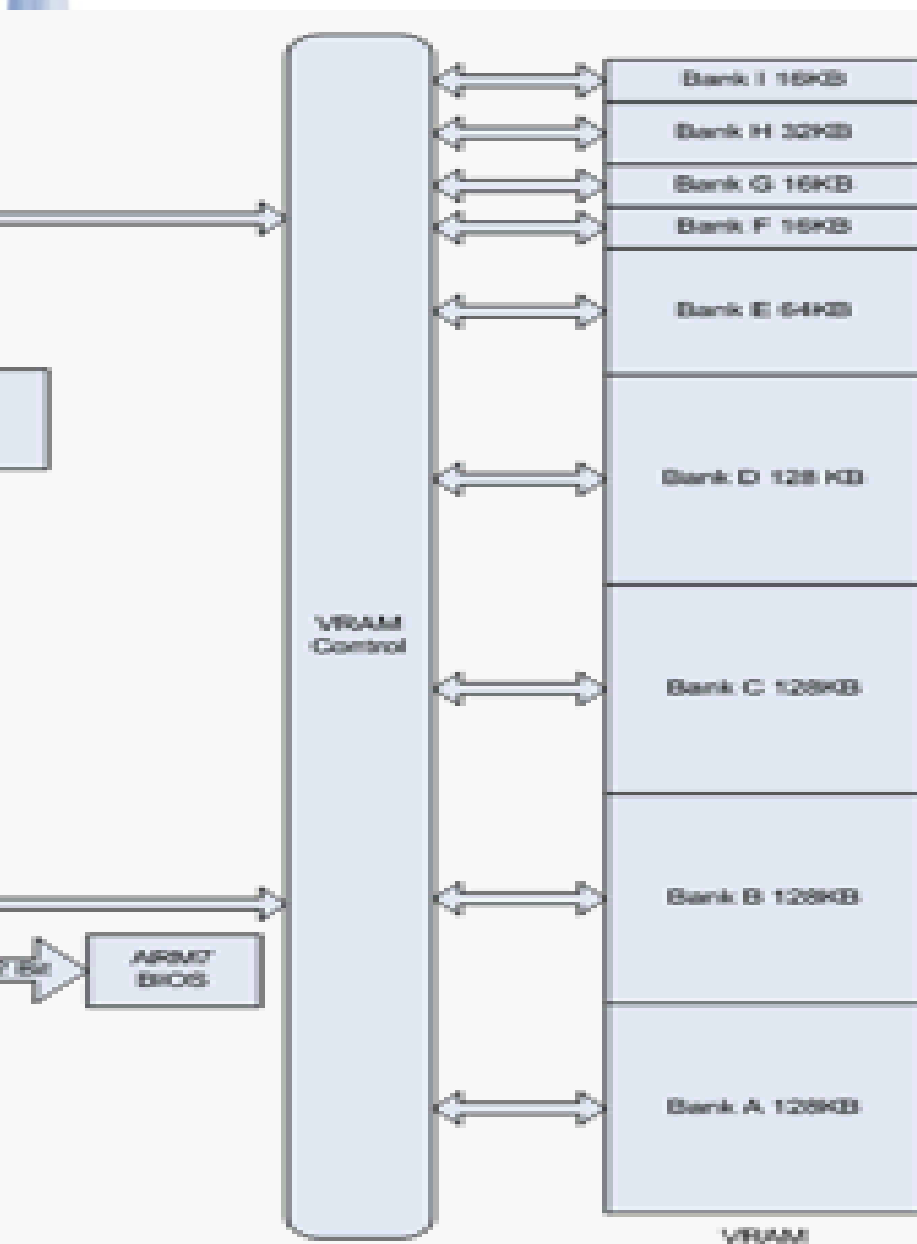
# Esquema de bloques

- *Memory layout*





# Esquema de bloques



res”

VRAM Memory Banks Bank name	Bank size (in Kilobytes)
VRAM_A	128
VRAM_B	128
VRAM_C	128
VRAM_D	128
VRAM_E	64
VRAM_F	16
VRAM_G	16
VRAM_H	32
VRAM_I	16
Total	656 KB

# VRAM

- *The 656 KB ← 9 bancos*
  - *Diferentes tamaños y propósitos*
  - *Se pueden asociar varios para un mismo propósito*
- *¿Qué bancos puede usar DS?*
  - *VRAM\_X\_TYPE: Main*
    - *VRAM\_X\_MAIN\_BG (X = A, B, C, D, E, F, G)*
    - *VRAM\_X\_MAIN\_SPRITE (X = A, B, C, D, E, F, G)*
  - *Sub*
  - *¿Comunes?*

# VRAM

- *¿Qué bancos puede usar DS? (II)*
  - *Main*
  - *VRAM\_X\_TYPE: Sub*
    - *VRAM\_X\_SUB\_BG (X= C, H, I)*
    - *VRAM\_X\_SUB\_SPRITE (X= D)*
    - *VRAM\_X\_SUB\_BG\_EXT\_PALETTE (X=H)*
    - *VRAM\_X\_SUB\_SPRITE\_EXT\_PALETTE (X=I)*
  - *Comunes*

# VRAM

- *¿Qué bancos puede usar DS? (y III)*
  - *Main*
  - *Sub*
  - *VRAM\_X\_TYPE: “comunes”*
    - *VRAM\_X\_LCD (X = A, B, C, D, E, F, G, H, I)*
      - *This is used for 3D and framebuffer mode.*
    - *VRAM\_X\_ARM7 (X= C, D)*
    - *VRAM\_X\_TEXTURE\_SLOTn*
      - *X= A,B,C, D n = 0, 1, 2, 3*
    - *VRAM\_X\_TEX\_PALETTE (X= E, F, G)*
    - *VRAM\_X\_BG\_EXT\_PALETTE (X= G, )*
    - *VRAM\_X\_OBJ\_EXT\_PALETTE (X=E, F, G)*

# Selección de bancos

- *Una ayuda: Bank selector*

– <http://mtheall.com/banks.html>

Function				Bank								
				A 128KB	B 128KB	C 128KB	D 128KB	E 64KB	F 16KB	G 16KB	H 32KB	I 16KB
LCD												
ARM7 Extra RAM (1st 128KB)												
ARM7 Extra RAM (2nd 128KB)												
Main 2D Engine	BG VRAM	1st 128KB	1st 16KB									
			2nd 16KB									
			3rd 16KB									
			4th 16KB									
			5th 16KB									
			6th 16KB									
			7th 16KB									
			8th 16KB									
		2nd 128KB	All									
		3rd 128KB	All									
		4th 128KB	All									
	OBJ VRAM	1st 128KB	1st 16KB									

– **Motor Main (I)**

Function call





















```
vramSetBankA(VRAM_A_MAIN_BG_0x06000000);
vramSetBankB(VRAM_B_LCD);
vramSetBankC(VRAM_C_LCD);
vramSetBankD(VRAM_D_LCD);
vramSetBankE(VRAM_E_LCD);
vramSetBankF(VRAM_F_LCD);
vramSetBankG(VRAM_G_LCD);
vramSetBankH(VRAM_H_LCD);
vramSetBankI(VRAM_I_LCD);
```

CPU Access

```
A: ARM9 0x06000000 - 0x0601FFFF (128KB)
B: ARM9 0x06820000 - 0x0683FFFF (128KB)
C: ARM9 0x06840000 - 0x0685FFFF (128KB)
D: ARM9 0x06860000 - 0x0687FFFF (128KB)
E: ARM9 0x06880000 - 0x0688FFFF ( 64KB)
F: ARM9 0x06890000 - 0x06893FFF ( 16KB)
G: ARM9 0x06894000 - 0x06897FFF ( 16KB)
H: ARM9 0x06898000 - 0x0689FFFF ( 32KB)
I: ARM9 0x068A0000 - 0x068A3FFF ( 16KB)
```

# Selección de bancos








- *Una ayuda: Bank selector*
  - <http://mtheall.com/banks.html>

	OBJ VRAM	1st 128KB	1st 16KB									
			2nd 16KB									
			3rd 16KB									
			4th 16KB									
			5th 16KB									
			6th 16KB									
			7th 16KB									
			8th 16KB									
		2nd 128KB	All 128KB									
	BG Ext Palette	Slot 0	8KB									
		Slot 1	8KB									
		Slot 2	8KB									
		Slot 3	8KB									
	OBJ Ext Palette	8KB										





























# Selección de bancos

- *Una ayuda: Bank selector*
  - *<http://mtheall.com/banks.html>*
  - *Motor Sub*

Sub 2D Engine	BG VRAM	128KB	1st 16KB									
			2nd 16KB									
			3rd 16KB									
			4th 16KB									
			5th 16KB									
			6th 16KB									
			7th 16KB									
			8th 16KB									
	OBJ VRAM	128KB	1st 16KB									
			Final 112KB									
	BG Ext Palette	32KB										
	OBJ Ext Palette	8KB										

# Selección de bancos

- Una ayuda: *Bank selector*
  - <http://mtheall.com/banks.html>
  - **Motor 3D**

3D Engine	Texture	Slot 0	128KB										
		Slot 1	128KB										
		Slot 2	128KB										
		Slot 3	128KB										
	Texture Palette	Slot 0	16KB										
		Slot 1	16KB										
		Slot 2	16KB										
		Slot 3	16KB										
		Slot 4	16KB										
		Slot 5	16KB										

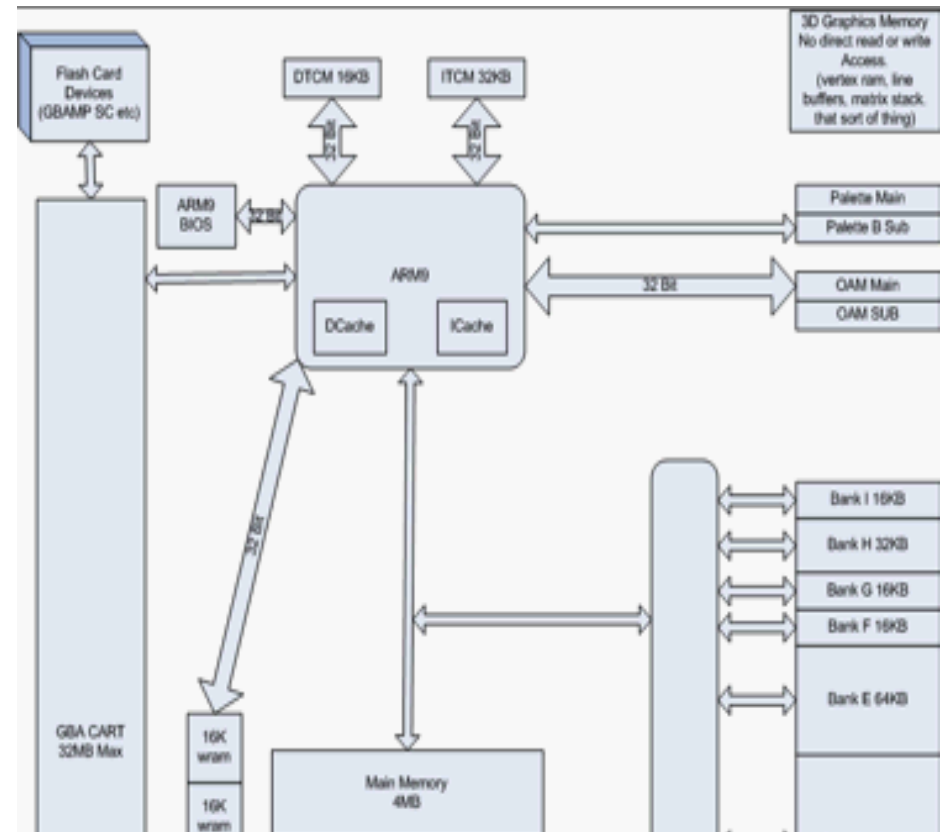
# Detección de conflictos

- Una ayuda: *VRAM BG Allocation Conflict Viewer*
  - *<http://mtheall.com/vram.html>*

[illegible]

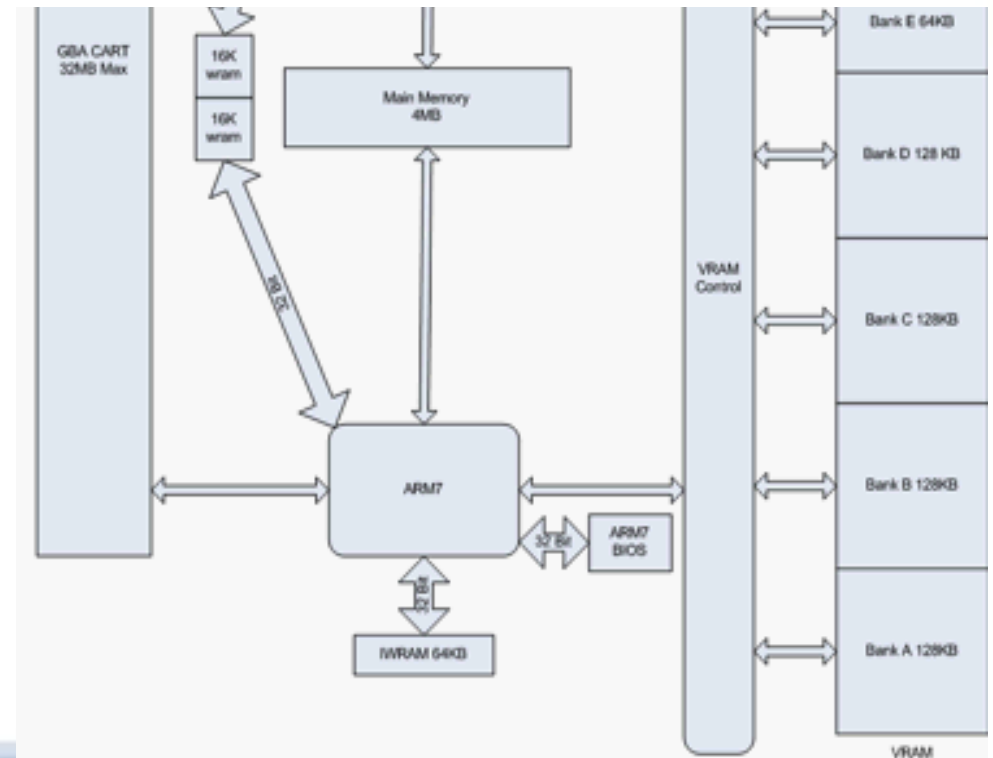
# Esquema de bloques (II)

- Elementos
  - ARM9 (66Mhz)
    - Lógica principal del programa



# Esquema de bloques (III)

- Elementos
  - ARM7 (33Mhz)
    - Audio
    - Red inalámbrica (*WiFi*)
    - Teclado



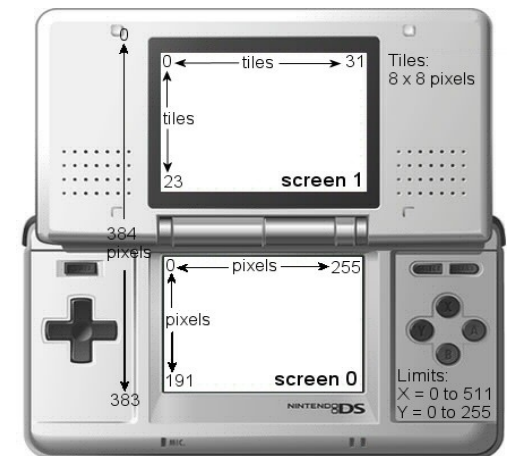
# Esquema de bloques (IV)

- RAM de 4 MB
  - Ejecutable para el ARM9
  - Datos globales de la aplicación
  - ARM9 y ARM7 pueden acceder
    - Si hay colisión, prioridad para el ARM7
- ¿MMU? ¿Contigua / Dispersa? ¿Páginas, marcos?
  - Asignación de bancos de memoria
    - Gestionada por la aplicación
    - *Ej.: DevkitPro* → código del ARM7
      - WRAM + IWRAM (96Kb = 32 + 46)



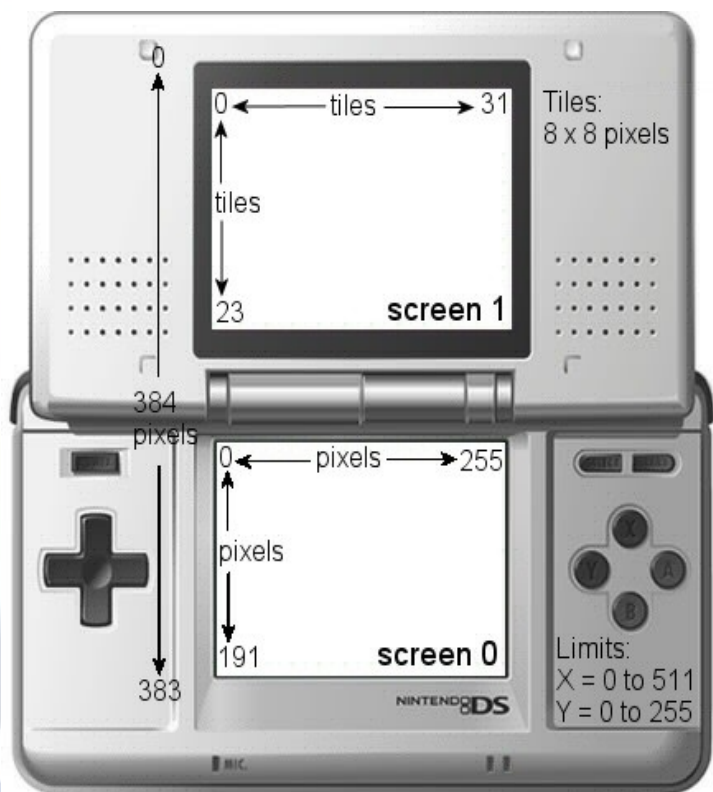
# Esquema de bloques (V)

- “Aceleradores”
  - 16 canales de sonido, micrófono y altavoces estéreo,
  - 2x4 temporizadores, aceleradores para las operaciones de división y raíz cuadrada, etc.
  - Modos gráficos
    - 2 motores gráficos 2D + 1 motor 3D
    - 2 pantallas: LCD + táctil
      - Resolución y modos gráficos



# Esquema de bloques (y VI)

- Modos gráficos 2D (motor MAIN y SUB)
  - Resolución NDS original (256x192), DSi
  - Resolución 3DS 400x240 (sup. 400xjo), 320x240 (inf.)



## Graphics Modes Main 2D Engine

Mode	BG0	BG1	BG2	BG3
Mode 0	Text/3D	Text	Text	Text
Mode 1	Text/3D	Text	Text	Rotation
Mode 2	Text/3D	Text	Rotation	Rotation
Mode 3	Text/3D	Text	Text	Extended
Mode 4	Text/3D	Text	Rotation	Extended
Mode 5	Text/3D	Text	Extended	Extended
Mode 6	3D	-	Large Bitmap	-
Frame Buffer	Direct VRAM display as a bitmap			

## Sub 2D Engine

Mode	BG0	BG1	BG2	BG3
Mode 0	Text	Text	Text	Text
Mode 1	Text	Text	Text	Rotation
Mode 2	Text	Text	Rotation	Rotation
Mode 3	Text	Text	Text	Extended
Mode 4	Text	Text	Rotation	Extended
Mode 5	Text	Text	Extended	Extended

# Arquitectura de la plataforma: el arranque

- El arranque de la consola
  - BIOS de la consola
    - Código de arranque de los dos procesadores
    - Lee una memoria flash de 256KiB
      - *Firmware* (cifrado) con el menú inicial y la aplicación *pictochat*
      - Lo copia (desencriptado) a la memoria principal y le da el control.
  - *Firmware* comprueba si el cartucho insertado es válido y lee las primeras posiciones de la memoria del cartucho.
    - Características del juego + posiciones códigos ARM9 y ARM7 y dónde se deben copiar.
    - Una vez copiados a memoria principal, el *firmware* les da el control.

# Arquitectura de la plataforma: el arranque

- El arranque de la consola (cont.)
  - BIOS de la consola, *firmware* ... cede el control al código del cartucho
  - El *firmware* no ejecutará programas que no estén debidamente firmados
- Técnicas para poder ejecutar soft. no oficial
  - *Pass-through*
    - Evitar tener que entender el mecanismo de cifrado.
    - Estas técnicas requieren un cartucho de Slot1 que simula el comportamiento de un cartucho de juego comercial.
  - *Flashme*
    - Reemplazar el *firmware* por una versión que no comprueba ningún tipo de firma.

# Arquitectura de la plataforma: el arranque

- Técnicas para poder ejecutar soft. no oficial
  - *Pass-through o flashme*
  - Los dispositivos (cartuchos)
    - Programa interno (navegador) que se comporta a su vez como cargador para otros programas.
    - El proceso de carga se produce de forma muy similar a como lo hace el firmware
      - El programa se lee de una tarjeta de memoria Flash utilizando un protocolo propio de cada fabricante
- SDK no oficial
  - “ROM” para el dispositivo de carga
  - Herramientas (DevkitPro)+ API básico (libnds)

# Arquitectura software: DevkitPro + libnds

- Para poder programar la consola
  - Crear uno de esos archivos (.nds) que emulan el contenido de la memoria ROM de un cartucho comercial.
- Para generar el ejecutable para la NDS
  - Si lenguaje C *arm-eabi-gcc / arm-eabi-gcc++*
    - Generación de código objeto para el ARM7 y ARM9:
  - Archivo ejecutable en un formato estándar: ELF
  - *Arm-eabi-objcopy* → generar los ejecutables reducidos *.arm7* y *.arm9*
  - *nds-tool* → cabecera + *.arm7* y *.arm9* + otros datos (p.ej. Gráficos) = *.nds*



# Arquitectura software: DevkitPro + libnds

- Entorno de desarrollo
  - DevkitPro
    - Conjunto de librerías, compiladores y utilidades que nos permitirán el desarrollo de aplicaciones para varios tipos de consolas, incluida la NDS.
    - Incluye *libnds*, una librería que se encarga de adaptar el código C al hardware específico de la NDS.
      - Michael Noland (joat), Jason Rogers (dovoto) y Dave Murphy (WinterMute)
  - Editor de textos: donde se escribirá el código de la aplicación.
  - Emulador, depurador, ...

# Arquitectura software: DevkitPro + libnds

- Estructura “general” de un desarrollo en *DevkitPro + libnds*
  - Directorio base del proyecto
    - *arm9 + arm7* ó *source + Makefile*
    - *data / audio* (XM, S3M, MOD), music (IT),
    - *data / gfx* (mapas de bits, instrucciones *grit*)
    - *build* (binarios intermedios ← *mmutil, bin2o, grit*)
    - *Makefile*
  - arm?
    - source
      - *main,c* ó *main.cpp*
    - *Makefile*

# Arquitectura software: DevkitPro + libnds

- Ejemplo de desarrollo

- Escoger una plantilla

```
$ cp -r ${DEVKITPRO}/examples/templates/arm9 hola
```

```
$ cd hola
```

- Editar el fichero *main.c*

```
#include <nds.h>
```

```
#include <stdio.h>
```

```
void main() {
```

```
    consoleDemoInit();
```

```
    printf("\n Hola, mundo\n");
```

```
}
```

- Compilación, prueba (emulación) y ejecución

```
$ make
```

```
$ desmume hola.nds
```

# Arquitectura software: DevkitPro + libnds

- API libnds:
  - Secciones en la documentación
    - *2D engine API*
    - *3D engine API*
    - *Audio AP*
    - *Memory*
    - *System*
    - *User Input/ouput*
    - *Utility*
    - *Custom Peripherals*
    - *Debugging*
  - Doxygen → gestión de la documentación
    - Local y en red <<http://libnds.devkitpro.org/index.html>>

# Arquitectura software: DevkitPro + libnds

- Algunas cuestiones de bajo nivel
  - Libnds. → ndstypes.h
    - double, float → float64, float32, int16, ...
    - byte, u8, int8, s8, vs8, ...
- Declaraciones y definiciones
  - define
  - aligned vs packed
  - const vs static
  - volatile vs register

# Estructura de una aplicación

- Estructura mínima
  - Dos versiones dentro de los ejemplos de *libnds*
    - v1  
    `${DEVKITPRO}/examples/nds/templates/arm9/source/main.c`
    - v2  
    `${DEVKITPRO}/examples/nds/hello_world/source/main.cpp`
  - videoSetMode, vramSetBankX, consoleInit
- Una aplicación compleja
  - DSOrganize

# Estructura de una aplicación

- Estructura mínima: v1

## main.c

```
/*-----  
    Basic template code for starting a DS app  
-----*/  
#include <nds.h>  
#include <stdio.h>  
//-----  
int main(void) {  
    //-----  
        consoleDemoInit();  
        iprintf("Hello World!");  
        while(1) {  
            swiWaitForVBlank();  
        }  
}
```



# Estructura de una aplicación

- Estructura mínima: v2

## main.cpp

```
/*-----  
-----  
    $Id: main.cpp,v 1.  
13 2008-12-02 20:21:20  
dovoto Exp $  
  
    Simple console print demo  
    -- dovoto  
-----  
-----*/  
#include <nds.h>  
#include <stdio.h>  
  
volatile int frame = 0;  
  
//-----  
-----  
void Vblank() {  
//-----  
-----  
    frame++;  
}  
""
```

```
...  
//-----  
int main(void) {  
//-----  
    touchPosition touchXY;  
  
    irqSet(IRQ_VBLANK, Vblank);  
  
    consoleDemolnit();  
  
    iprintf("  Hello DS dev'rs\n");  
    iprintf("  \x1b[32mwww.devkitpro.org\n");  
    iprintf("  \x1b[32;1mwww.drunkenoders.com\x1b[39m");  
  
    while(1) {  
  
        swiWaitForVBlank();  
        touchRead(&touchXY);  
  
        // print at using ansi escape sequence \x1b[line;columnH  
        iprintf("\x1b[10;0HFrame = %d",frame);  
        iprintf("\x1b[16;0HTouch x = %04X, %04X\n",  
                touchXY.rawx, touchXY.px);  
        iprintf("Touch y = %04X, %04X\n", touchXY.rawy, touchXY.py);  
    }  
    return 0;  
}
```

# consoleDemolnit

- Fichero:  
\${DEVKITPRO}/libnds-src-1.5.12/source/arm9/console.c

```
//-----  
// Places the console in a default mode using bg0 of the sub display, and vram c for  
// font and map..this is provided for rapid prototyping and nothing more  
PrintConsole* consoleDemolnit(void) {  
//-----  
    videoSetModeSub(MODE_0_2D);  
    vramSetBankC(VRAM_C_SUB_BG);  
  
    return consoleInit(NULL, defaultConsole.bgLayer, BgType_Text4bpp, BgSize_T_256x256,  
        defaultConsole.mapBase, defaultConsole.gfxBase, false, true);  
}
```

# Modos gráficos

- Enumerados para los modos gráficos
  - MAIN
  - SUB

Enumerator	
<i>MODE_0_2D</i>	4 2D backgrounds
<i>MODE_1_2D</i>	4 2D backgrounds
<i>MODE_2_2D</i>	4 2D backgrounds
<i>MODE_3_2D</i>	4 2D backgrounds
<i>MODE_4_2D</i>	4 2D backgrounds
<i>MODE_5_2D</i>	4 2D backgrounds
<i>MODE_6_2D</i>	4 2D backgrounds
<i>MODE_0_3D</i>	3 2D backgrounds 1 3D background (Main engine only)
<i>MODE_1_3D</i>	3 2D backgrounds 1 3D background (Main engine only)
<i>MODE_2_3D</i>	3 2D backgrounds 1 3D background (Main engine only)
<i>MODE_3_3D</i>	3 2D backgrounds 1 3D background (Main engine only)
<i>MODE_4_3D</i>	3 2D backgrounds 1 3D background (Main engine only)
<i>MODE_5_3D</i>	3 2D backgrounds 1 3D background (Main engine only)
<i>MODE_6_3D</i>	3 2D backgrounds 1 3D background (Main engine only)
<i>MODE_FIFO</i>	video display from main memory
<i>MODE_FB0</i>	video display directly from VRAM_A in LCD mode
<i>MODE_FB1</i>	video display directly from VRAM_B in LCD mode
<i>MODE_FB2</i>	video display directly from VRAM_C in LCD mode
<i>MODE_FB3</i>	video display directly from VRAM_D in LCD mode

# Inicialización gráfica

- First Demo: NDS/Tutorials Day 1 - Dev-Scene

```
#include <nds.h>
#include <stdio.h>

int main(void)
{
    int i;

    consoleDemoInit();
    videoSetMode(MODE_FB0);
    vramSetBankA(VRAM_A_LCD);

    printf("Hello World!\n");
    printf("www.Drunkencoders.com");

    for(i = 0; i < 256 * 192; i++)
        VRAM_A[i] = RGB15(31,0,0);

    while(1){
        swiWaitForVBlank();
    }

    return 0;
}
```



# Inicialización gráfica

- How do I draw to the screen as a 16 bit bitmap?

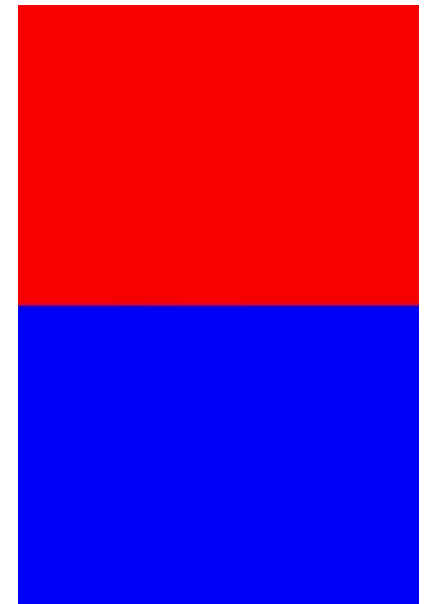
```
//set the mode to allow for an extended rotation background
videoSetMode(MODE_5_2D);
videoSetModeSub(MODE_5_2D);

//allocate a vram bank for each display
vramSetBankA(VRAM_A_MAIN_BG);
vramSetBankC(VRAM_C_SUB_BG);

//create a background on each display
int bgMain = bgInit(3, BgType_Bmp16, BgSize_B16_256x256, 0,0);
int bgSub = bgInitSub(3, BgType_Bmp16, BgSize_B16_256x256, 0,0);

u16* videoMemoryMain = bgGetGfxPtr(bgMain);
u16* videoMemorySub = bgGetGfxPtr(bgSub);

//initialize it with a color
for(x = 0; x < 256; x++)
    for(y = 0; y < 256; y++)
    {
        videoMemoryMain[x + y * 256] = ARGB16(1, 31, 0, 0);
        videoMemorySub[x + y * 256] = ARGB16(1, 0, 0, 31);
    }
```





# Inicialización gráfica

- How do I draw to the screen as a 16 bit bitmap?

...

En general				
			bits	Kb
256	192	15	737280	720
			Bytes	KB
			92160	90
BgSize_B16_256_256			bits	Kb
256	256	16	1048576	1024
			Bytes	KB
			131072	128

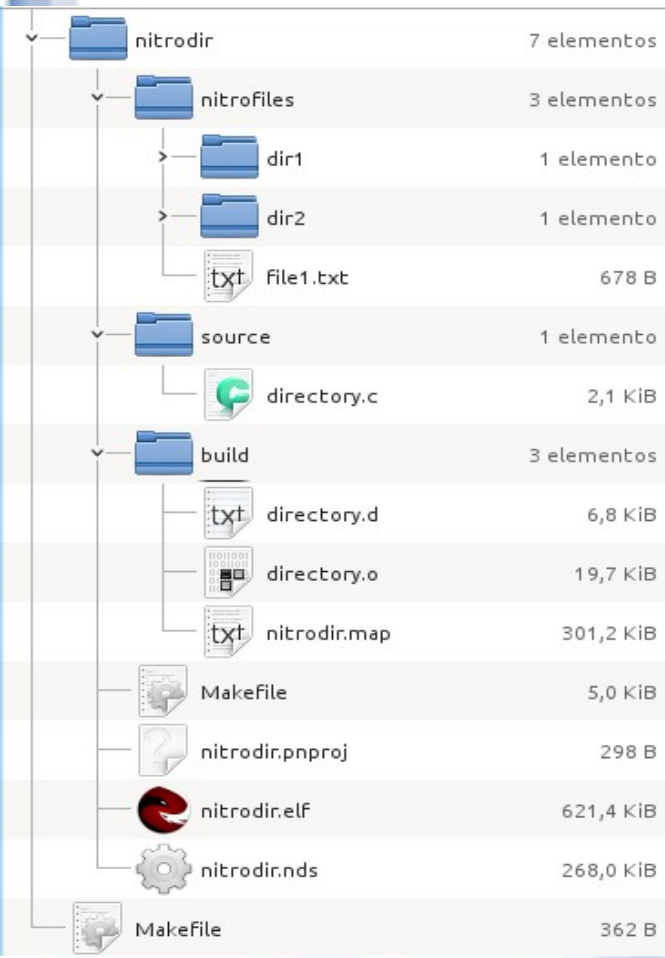
# Acceso al sistema de ficheros

- NitroFS
  - Sistema de solo lectura “incrustado” en el NDS
  - No es visible por el usuario del NDS
- FAT
  - Acceso r/w al sistema de ficheros de la tarjeta de memoria del cartucho (*flashcard*)
  - Versiones FAT: 12, 16, 32

# Acceso al sistema de ficheros

- NitroFS

– `${DEVKITPRO}/examples/nds/filesystem/nitrofs`



```
int main(int argc, char **argv) {
    // Initialise the console, required for printf
    consoleDemolnit();

    if (nitroFSInit(NULL)) {

        dirlist("/");

        ...

        FILE* inf = fopen("file1.txt", "rb");

        ...

        fseek(inf, 0, SEEK_END);

        ...

        if (fread(entireFile, 1, len, inf) != len)

        ...

        fclose(inf);

        ...
    } else {
        iprintf("nitroFSInit failure: terminating\n");
    }

    while(1) {
        swiWaitForVBlank();
        scanKeys();
        if (keysDown() & KEY_START) break;
    }

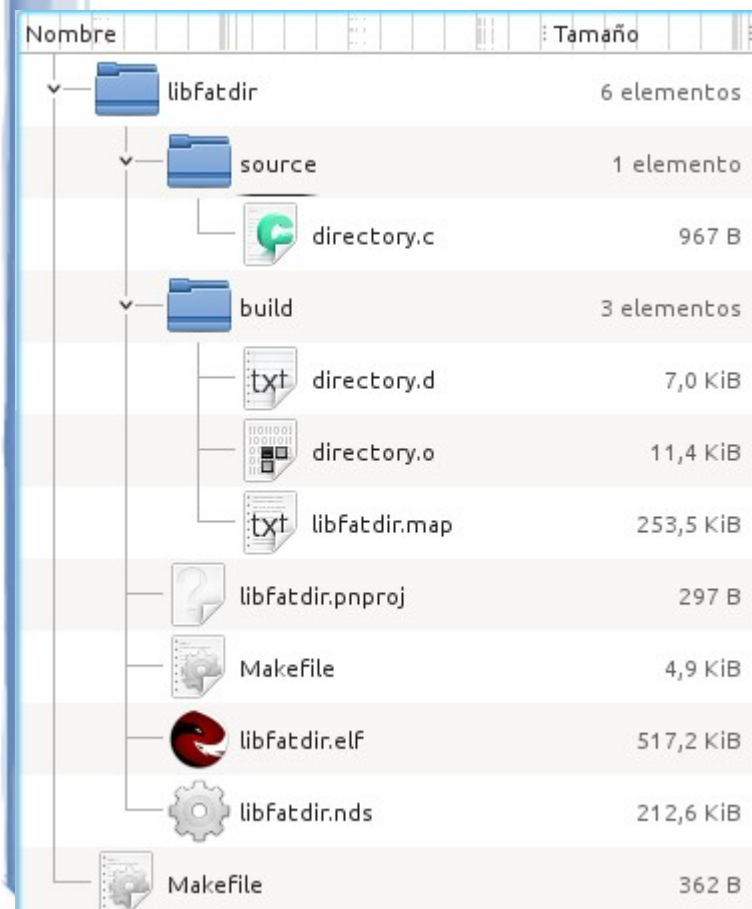
    return 0;
}
```



# Acceso al sistema de ficheros

- FAT

– `${DEVKITPRO}/examples/nds/filesystem/libfat`



```
int main(int argc, char **argv) {
    // Initialise the console, required for printf
    consoleDemolnit();

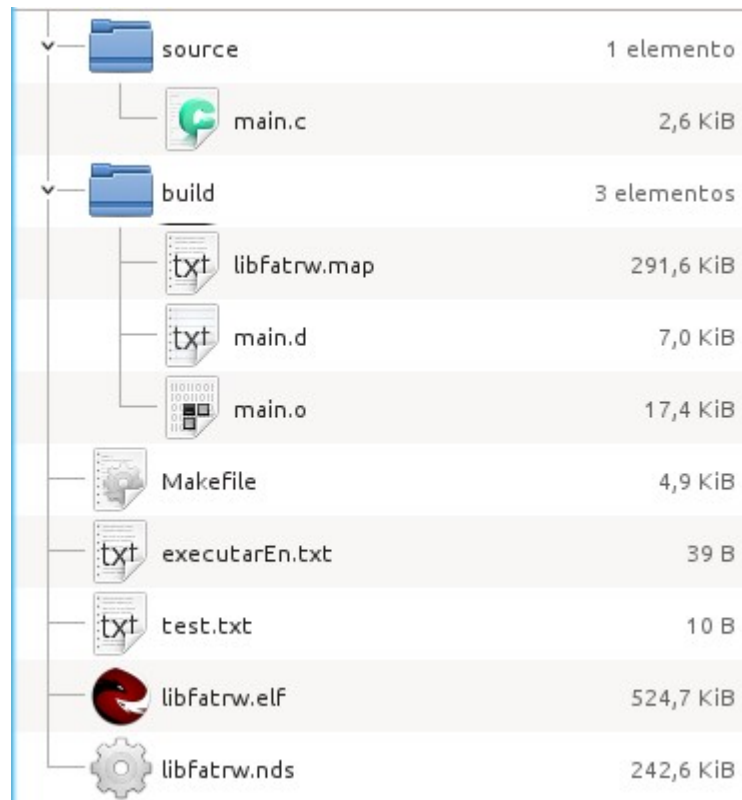
    if (fatInitDefault()) {
        DIR *pdir;
        struct dirent *pent;
        pdir=opendir("/");
        if (pdir){
            while ((pent=readdir(pdir))!=NULL) {
                if(strcmp(".", pent->d_name) == 0 ||
                   strcmp("..", pent->d_name) == 0)
                    continue;
                if(pent->d_type == DT_DIR)
                    iprintf("[%s]\n", pent->d_name);
                else
                    iprintf("%s\n", pent->d_name);
            }
            closedir(pdir);
        } else {
            iprintf ("opendir() failure; terminating\n");
        }
    } else {
        iprintf("fatInitDefault failure: terminating\n");
    }

    while(1) { swiWaitForVBlank(); }
    return 0;
}
```



# Acceso al sistema de ficheros

- Caso de ejemplo sobre DeSmuME
  - Contenido de un directorio en disco (.../libfatrw)



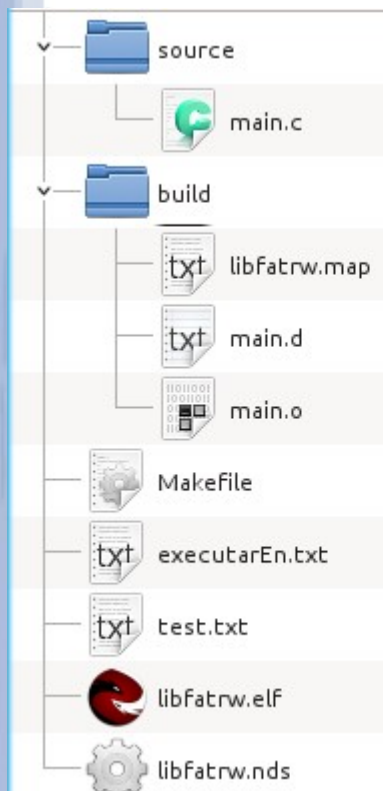
source	1 elemento
main.c	2,6 KiB
build	3 elementos
libfatrw.map	291,6 KiB
main.d	7,0 KiB
main.o	17,4 KiB
Makefile	4,9 KiB
executarEn.txt	39 B
test.txt	10 B
libfatrw.elf	524,7 KiB
libfatrw.nds	242,6 KiB



# Acceso al sistema de ficheros

- Caso de ejemplo sobre DeSmuME

– \$ desmume --cflash-path=./ libfatrw.nds → **IMPTE.:** --cflash-path=./  
Nintendo DS rom tool 1.50.1 - Jun 19 2012



...  
Using CFlash directory: ./  
cflash added main.original  
cflash added libfatrw.nds  
cflash added Makefile  
cflash added source  
cflash added main.c  
cflash added ..

→ **crea un sistema de archivos temporal**

→ **entra recursivamente en directorios**

→ **vuelve al directorio padre y continua**

...  
cflash added test.txt

...  
Trying with 1 sectors/cluster: → **decide el tipo de sistema de archivos**

FAT12: #clu=73266, fatlen=215, maxclu=4080, limit=4080

FAT12: too much clusters

FAT16: #clu=73124, fatlen=286, maxclu=65520, limit=65520

FAT16: too much clusters

FAT16: would be misdetected as FAT12

FAT32: #clu=72562, fatlen=567, maxclu=72576, limit=268435440

Using sector 6 as backup boot sector (0 = none)

...  
DeSmuME .dsv save file not found. Trying to load an old raw .sav file.

Missing save file /home/magusti/.config/desmume/libfatrw.dsv

\$.

# Acceso al sistema de ficheros

- Caso de ejemplo sobre DeSmuME

- Código fuente

```
#include <nds.h>
#include <fat.h>
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
```

```
int main(int argc, char **argv) {
    FILE *fp;
    u8 c;
    consoleDemolnit();
    printf("Init FAT: fatInitDefault!\n");
    if(!fatInitDefault())
        printf("Init FAT: Error!\n");
    else
    {
        printf("Init FAT: conseguit!\n");
        // Ejemplo de acceso en modo texto: lectura
        printf("Operaciones en modo texto\nLlegint fat:/test.txt\n");
        fp = fopen("fat:/test.txt", "r");
        if(!fp)
            printf("Llegint fat:/test.txt: Error!\n");
        else
        {
            while (!feof( fp )) {
                c = fgetc(fp);
                printf("%c", c);
            }
            printf("\n");
            fclose(fp);
        }
    }
}
```

...

...  
// Ejemplo de acceso en modo texto: escritura

```
printf("Escribint fat:/test.txt\n");
fp = fopen("fat:/test.txt", "w");
if(!fp)
    printf("Escribint fat:/test.txt: Error!\n");
else
{
    fprintf(fp, "%s\n", "Manolo" );
    printf("\n");
    fclose(fp);
}
```

// Comprobación

// Recordar que al salir de desmume no permanecen los cambios

```
fp = fopen("fat:/test.txt", "r");
if(!fp)
    printf("Tornant a llegir fat:/test.txt: Error!\n");
else
{
    while (!feof( fp )) {
        c = fgetc(fp);
        printf("%c", c);
    }
    printf("\n");
    fclose(fp);
}
```

...

# Acceso al sistema de ficheros

- Caso de ejemplo sobre DeSmuME

- Código fuente (y II)

```
printf("Init FAT: fatInitDefault!\n");

if(!fatInitDefault())
    printf("Init FAT: Error!\n");
else
{
    ...
    // Acceso en modo binario
    fp = fopen("fat:/Tetris.sav", "wb");
    if(!fp) printf("Writing fat:/Tetris.sav: Error!\n");
    else
    {
        printf("Write something to fat:/Tetris.sav\n");
        fputc(0x78, fp);
        fclose(fp);
    }

    fp = fopen("fat:/Tetris.sav", "rb");
    if(!fp) printf("Reading fat:/Tetris.sav: Error!\n");
    else
    {
        u8 c = fgetc(fp);
        fclose(fp);
        printf("Char read: 0x%02X\n", c);
    }

}

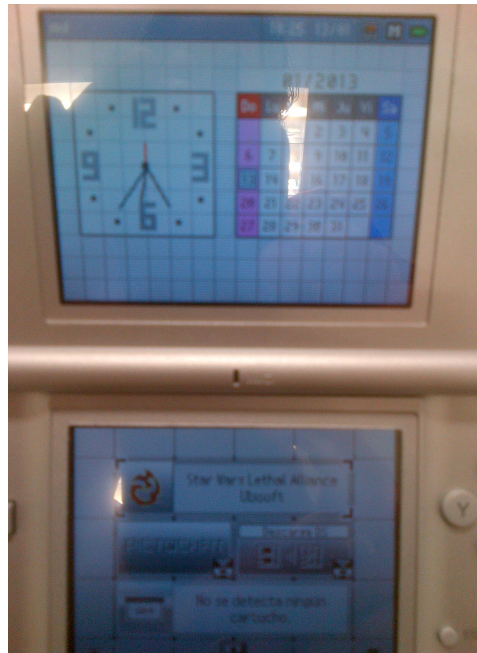
while(1) { swiWaitForVBlank(); }
return 0;
}
```

# En prácticas

- R4 en Slot-1 NDS



- R4i-SDHC se identifica como “Star Wars Lethal Alliance Ubisoft”





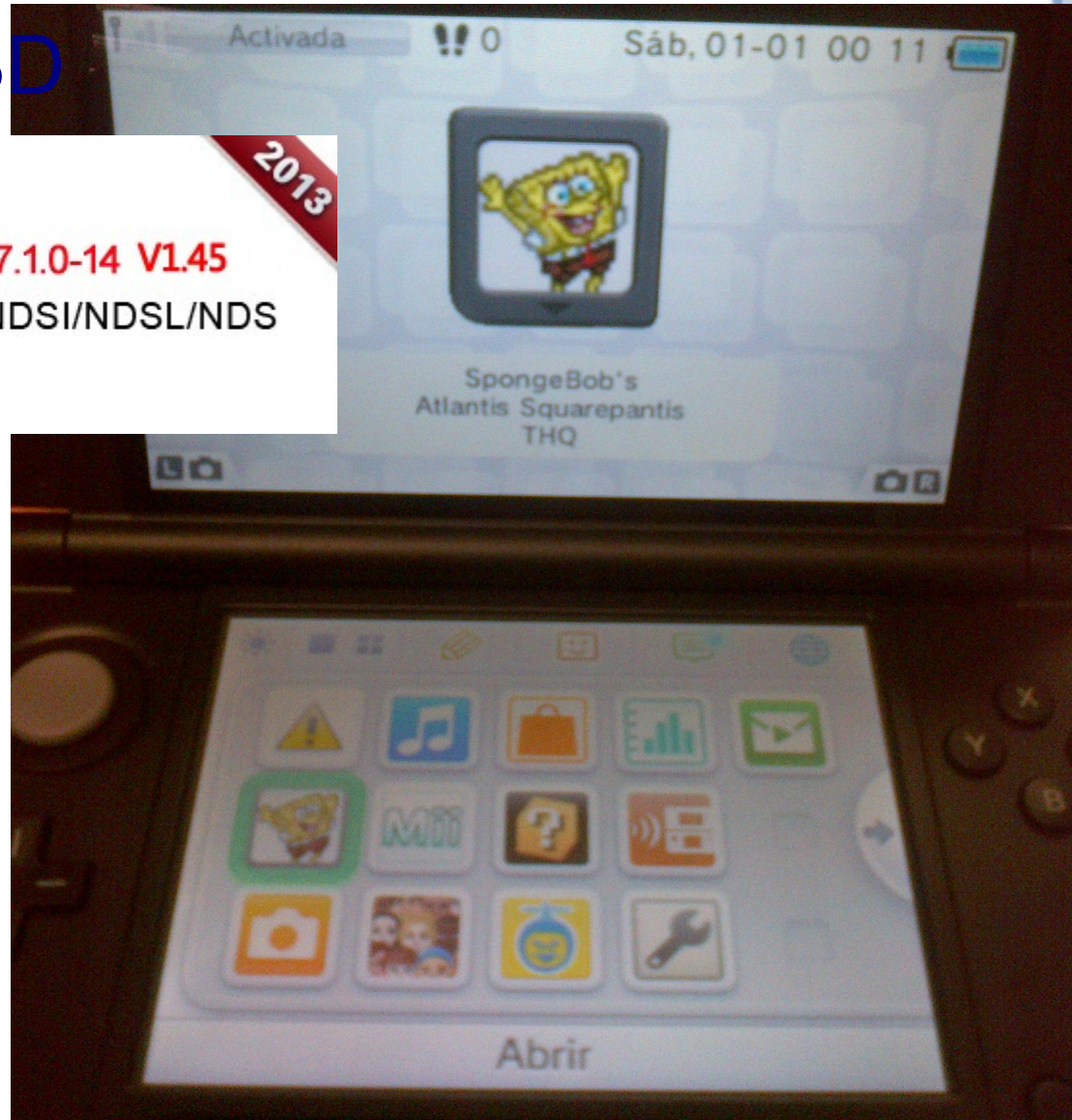
# En prácticas

- R4 en Slot-1 NDS 3D



**Compatible** V7.1.0-14 V1.45  
3DS/NDSILL[XL]/NDSI/NDL/NDL

- *R4 Gold Pro*  
se identifica  
como  
“Sponge  
Bob's  
Atlantis  
Squarepantis  
THQ”



# En prácticas

- Acceso a datos “personales”
  - tPERSONAL\_DATA
    - [<http://libnds.devkitpro.org/a00098.html>](http://libnds.devkitpro.org/a00098.html)
  - The personal data structure is at a specific address, the code you show here merely reserves memory for a struct of that type and does not initialise it.

```
int myLanguage = PersonalData->language;
```

- With the next libnds update we have been getting rid of anonymous structs and unions for C99 compatibility, once that is released you would use

```
int myLanguage = PersonalData->_userdata.language;
```



# Bibliografía

- Herramientas de desarrollo
  - DevkitPro
  - libnds
- Descripción de la plataforma
  - Nintendo DS homebrew - Wikipedia
  - Nintendo DS
- Foros de desarrollo
  - DevScene
  - *Drunken Coders* <<http://drunkencoders.com>>
  - 3DBrew <<http://3dbrew.org/wiki>>

## Bibliografía (II)

- Jaeden Amero (Patater). 2010. Introduction to Nintendo DS Programming

— [<http://www.patater.com>](http://www.patater.com)

- 3DGuy. 3DS Development Hardware
- Neimod y Martin Korth. 2013. DSTek: Nintendo DS Technical Information
- GbaTek. Technical information from no\$gba
- Märten Tonisoo. 2010. Nintendo DS game console.

— [<http://www.martentonisoo.com/documents/Nintendo\\_DS\\_game\\_console.pdf>](http://www.martentonisoo.com/documents/Nintendo_DS_game_console.pdf)

- ThomasWorld. Baby Steps In Nintendo DS Homebrew Hacking

## Bibliografía (III)

- F. García Bernal. 2008. Desarrollo de Videojuego 3D Para La Videoconsola Nintendo DS.
  - Dpto. Lenguajes y Ciencias de la Computación. ETS ING.INF. Universidad de Málaga.
- Puyover. 2010. Programación en Libnds y ensamblador ARM. Una introducción a la programación en NDS
  - [http://www.martentonissou.com/documents/Nintendo\\_DS\\_game\\_console.pdf](http://www.martentonissou.com/documents/Nintendo_DS_game_console.pdf)
- F. Sivianes, J. Barros y A. Martín. 2009. Entorno Desarrollo para NDS.
  - Periféricos e Interfaces. 3º Ingeniería Técnica en Informática, especialidad en Sistemas Físicos
  - [http://www.dte.us.es/tec\\_inf/itis/peri\\_int/EvolucionInicio/Trabajo\\_NDS.pdf](http://www.dte.us.es/tec_inf/itis/peri_int/EvolucionInicio/Trabajo_NDS.pdf)

# Bibliografía (y IV)

- M. Banahan, D. Brady y M. Doran. 1991. The C Book. Addison Wesley.

– En línea

[<http://publications.gbdirect.co.uk/c\\_book/chapter8/declarations\\_and\\_definitions.html>](http://publications.gbdirect.co.uk/c_book/chapter8/declarations_and_definitions.html)

- B, W. Kernighan y R. Pike. 1983. The Unix Programming Environment. Prentice-Hall Software Series.

by Brian W. Kernighan (Author) , Rob Pike

- B. W. Kernighan y D. M. Ritchie. 1998. C Programming Language (2nd Edition)