UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

CARRERA DE ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

RETRO-CIAA: Consola de videojuegos basada en EDU-CIAA-NXP

Autor:
Lic. Santiago Germino

Director:
Ing. Gustavo Alessandrini (INTI, ORT)

Jurados:
Ing. Fernando Lichtschein (FIUBA)
Esp. Ing. Patricio Bos (FIUBA)
Esp. Ing. Pablo O. Ridolfi (UTN-FRBA, FIUBA)

Este trabajo fue realizado en las Ciudad Autónoma de Buenos Aires, entre mayo de 2018 y diciembre de 2018.

Resumen

La presente memoria describe el diseño e implementación de una placa de expansión y firmware que suma capacidades de conexión inalámbrica, audio y video de alta performance a la EDU-CIAA-NXP (versión educativa del proyecto CIAA - Computadora Industrial abierta Argentina).

La EDU-CIAA-NXP no ha sido diseñada para aplicaciones multimedia y los recursos disponibles son escasos. Con esta expansión se obtiene video fluido pero de baja resolución para los estándares actuales. Para transformar esta limitación en un atractivo, se imita el concepto de las consolas retro (de estilo antiguo) que poseen una resolución de video similar y un estilo de imagen inconfundible. De ahí surgió el nombre de esta expansión multimedia: RETRO-CIAA.

El objetivo es brindar una herramienta atractiva para impulsar la practica de *Computación Gráfica* en el país, esto es, la generación de imágenes por computadora; disciplina que posibilitó el desarrollo de los videojuegos, los efectos especiales por computadora en películas y las interfaces gráficas de usuario (GUI) entre otras innovaciones.

Agradecimientos

Al amor incondicional de mis padres y su visión de acercarme una PC a muy corta edad. A los videojuegos de los 80' y las revistas «gallegas» de los 90'. A mi compañero de aventuras informáticas y a los locales adonde íbamos a grabar juegos en disquetes de 5 ¼. A mi hermano por haber sido un gamer mucho mas dedicado que yo y compartir tantas horas de juego conmigo. A mi tio, que reconoció la importancia de enseñar informática y robótica varias décadas antes de que eso fuese urgente. A la Demoscene mundial por mezclar arte con técnica y brindarme una motivación infinita para encarar el camino de la programación gráfica y de bajo nivel.

Al Director de este trabajo Ing. Gustavo Alessandrini por su ejemplo profesional, rigurosidad y sugerencias siempre acertadas.

Al Dr. Ing. Ariel Lutemberg por su empuje incesante y su aporte invaluable al desarrollo de los sistemas embebidos en el país.

A mis profesores de la especialización por el conocimiento brindado y la excelente predisposición.

A mis compañeros por toda su ayuda y la muy grata compañía a lo largo de la cursada.

Este trabajo no hubiese sido posible sin estas y muchas otras personas que aparecieron en mi vida en el momento oportuno. Así que, a todos ellos, muchísimas gracias.

Índice general

Índice de figuras

Índice de Tablas

Índice de Listados

Dedicado a los ya extintos salones de videojuegos o «Arcades». Los voy a extrañar.

Licencias

A continuación se listan las diferentes licencias de uso y distribución del trabajo contenido en esta memoria.

El código fuente se distribuye bajo licencia *BSD 3-clause*.

La documentación sobre el diseño de hardware de la RETRO-CIAA -esquemáticos, diagramas y explicaciones sobre su funcionamiento- se distribuye bajo licencia *CERN Open Hardware Licence v1.2*.

```
Documentation means schematic diagrams, designs, circuit or circuit
board layouts, mechanical drawings, flow charts and descriptive text,
and other explanatory material that is explicitly stated as being made
available under the conditions of this Licence. The Documentation may
be in any medium, including but not limited to computer files and
representations on paper, film, or any other media.

Documentation Location means a location where the Licensor has
placed Documentation, and which he believes will be publicly
accessible for at least three years from the first communication to
the public or distribution of Documentation.

Product means either an entire, or any part of a, device built using
the Documentation or the modified Documentation.

Licensee means any natural or legal person exercising rights under
this Licence.

Licensor means any natural or legal person that creates or modifies
Documentation and subsequently communicates to the public and/ or
distributes the resulting Documentation under the terms and conditions
of this Licence.

A Licensee may at the same time be a Licensor, and vice versa.

Use of the masculine gender includes the feminine and neuter genders
and is employed solely to facilitate reading.

2. Applicability

2.1. This Licence governs the use, copying, modification,
communication to the public and distribution of the Documentation, and
the manufacture and distribution of Products. By exercising any right
granted under this Licence, the Licensee irrevocably accepts these
terms and conditions.

2.2. This Licence is granted by the Licensor directly to the Licensee,
and shall apply worldwide and without limitation in time. The Licensee
may assign his licence rights or grant sub-licences.

2.3. This Licence does not extend to software, firmware, or code
loaded into programmable devices which may be used in conjunction with
the Documentation, the modified Documentation or with Products, unless
such software, firmware, or code is explicitly expressed to be subject
to this Licence. The use of such software, firmware, or code is
otherwise subject to the applicable licence terms and conditions.

3. Copying, modification, communication to the public and distribution
of the Documentation

3.1. The Licensee shall keep intact all copyright and trademarks
notices, all notices referring to Documentation Location, and all
notices that refer to this Licence and to the disclaimer of warranties
that are included in the Documentation. He shall include a copy
thereof in every copy of the Documentation or, as the case may be,
modified Documentation, that he communicates to the public or
distributes.

3.2. The Licensee may copy, communicate to the public and distribute
verbatim copies of the Documentation, in any medium, subject to the
requirements specified in section 3.1.

3.3. The Licensee may modify the Documentation or any portion thereof
provided that upon modification of the Documentation, the Licensee
shall make the modified Documentation available from a Documentation
Location such that it can be easily located by an original Licensor
once the Licensee communicates to the public or distributes the
modified Documentation under section 3.4, and, where required by
section 4.1, by a recipient of a Product. However, the Licensor shall
not assert his rights under the foregoing proviso unless or until a
Product is distributed.
```

```
3.4. The Licensee may communicate to the public and distribute the
modified Documentation (thereby in addition to being a Licensee also
becoming a Licensor), always provided that he shall:

a) comply with section 3.1;

b) cause the modified Documentation to carry prominent notices stating
that the Licensee has modified the Documentation, with the date and
description of the modifications;

c) cause the modified Documentation to carry a new Documentation
Location notice if the original Documentation provided for one;

d) make available the modified Documentation at the same level of
abstraction as that of the Documentation, in the preferred format for
making modifications to it (e.g. the native format of the CAD tool as
applicable), and in the event that format is proprietary, in a format
viewable with a tool licensed under an OSI-approved license if the
proprietary tool can create it; and

e) license the modified Documentation under the terms and conditions
of this Licence or, where applicable, a later version of this Licence
as may be issued by CERN.

3.5. The Licence includes a non-exclusive licence to those patents or
registered designs that are held by, under the control of, or
sub-licensable by the Licensor, to the extent necessary to make use of
the rights granted under this Licence. The scope of this section 3.5
shall be strictly limited to the parts of the Documentation or
modified Documentation created by the Licensor.

4. Manufacture and distribution of Products

4.1. The Licensee may manufacture or distribute Products always
provided that, where such manufacture or distribution requires a
licence under this Licence the Licensee provides to each recipient of
such Products an easy means of accessing a copy of the Documentation
or modified Documentation, as applicable, as set out in section 3.

4.2. The Licensee is invited to inform any Licensor who has indicated
his wish to receive this information about the type, quantity and
dates of production of Products the Licensee has (had) manufactured

5. Warranty and liability

5.1. DISCLAIMER  The Documentation and any modified Documentation are
provided "as is" and any express or implied warranties, including, but
not limited to, implied warranties of merchantability, of satisfactory
quality, non-infringement of third party rights, and fitness for a
particular purpose or use are disclaimed in respect of the
Documentation, the modified Documentation or any Product. The Licensor
makes no representation that the Documentation, modified
Documentation, or any Product, does or will not infringe any patent,
copyright, trade secret or other proprietary right. The entire risk as
to the use, quality, and performance of a Product shall be with the
Licensee and not the Licensor. This disclaimer of warranty is an
essential part of this Licence and a condition for the grant of any
rights granted under this Licence. The Licensee warrants that it does
not act in a consumer capacity.

5.2. LIMITATION OF LIABILITY  The Licensor shall have no liability
for direct, indirect, special, incidental, consequential, exemplary,
punitive or other damages of any character including, without
limitation, procurement of substitute goods or services, loss of use,
data or profits, or business interruption, however caused and on any
theory of contract, warranty, tort (including negligence), product
liability or otherwise, arising in any way in relation to the
Documentation, modified Documentation and/or the use, manufacture or
distribution of a Product, even if advised of the possibility of such
damages, and the Licensee shall hold the Licensor(s) free and harmless
from any liability, costs, damages, fees and expenses, including
claims by third parties, in relation to such use.
```

```
6. General

6.1. Except for the rights explicitly granted hereunder, this Licence
does not imply or represent any transfer or assignment of intellectual
property rights to the Licensee.

6.2. The Licensee shall not use or make reference to any of the names
(including acronyms and abbreviations), images, or logos under which
the Licensor is known, save in so far as required to comply with
section 3. Any such permitted use or reference shall be factual and
shall in no event suggest any kind of endorsement by the Licensor or
its personnel of the modified Documentation or any Product, or any
kind of implication by the Licensor or its personnel in the
preparation of the modified Documentation or Product.

6.3. CERN may publish updated versions of this Licence which retain
the same general provisions as this version, but differ in detail so
far this is required and reasonable. New versions will be published
with a unique version number.

6.4. This Licence shall terminate with immediate effect, upon written
notice and without involvement of a court if the Licensee fails to
comply with any of its terms and conditions, or if the Licensee
initiates legal action against Licensor in relation to this
Licence. Section 5 shall continue to apply.
```

Convenciones

Para todo uso se considera que un *byte* = $8\,\mathrm{bit}$.

Una cantidad de *bytes* se expresa como $1234\,\mathrm{B}$.

Los tamaños de memoria son magnitudes base 2. $1024\,\mathrm{B} = 1\,\mathrm{KiB}$.

Los números en hexadecimal se representan en formato $\mathrm{A0\,B1\,C2\,D3\,h}$.

Puede indicarse numero de página en la referencia bibliográfica [3, p.250].

Capítulo 1

Introducción General

1.1. Contexto

El presente trabajo consiste en el diseño e implementación de una expansión para una placa base que tiene su propia historia, características y objetivos. En esta sección se presentará dicha placa base: la EDU-CIAA-NXP desarrollada y promovida por el Proyecto CIAA.

1.1.1. Proyecto CIAA

El Proyecto CIAA (Computadora Industria Abierta Argentina) surgió en el año 2013 como una iniciativa conjunta entre sectores industriales y académicos. Los objetivos generales son:

- Impulsar el desarrollo tecnológico nacional, a partir de sumar valor agregado al trabajo y a los productos y servicios, mediante el uso de sistemas electrónicos, en el marco de la vinculación de las instituciones educativas y el sistema científico-tecnológico con la industria.
- Darle visibilidad positiva a la electrónica Argentina.
- Generar cambios estructurales en la forma en la que se desarrollan y utilizan en la Argentina los conocimientos de la electrónica en la industria.

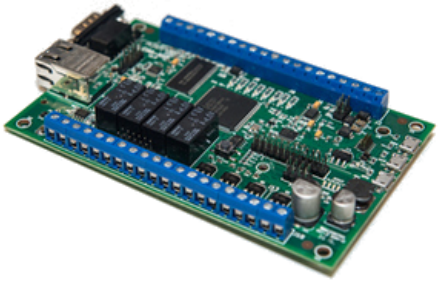


FIGURA 1.1: Computadora Industrial Abierta Argentina. Modelo CIAA-NXP.

Bajo este proyecto se creo la plataforma CIAA, una computadora diseñada para controlar procesos industriales. También se elaboro el firmware[1]y ejemplos de aplicación. Todo el trabajo es de código y diseño abierto.

Actualmente existen dos variantes de CIAA según la marca del microcontrolador que incorpora: CIAA-NXP con microcontrolador de NXP Semiconductors N.V. y CIAA-FSL que integra un microcontrolador de FreeScale Semiconductor Inc.[2]

1.1.2. EDU-CIAA

Las placas EDU-CIAA están diseñadas para entornos educativos y son de bajo costo. Carecen de todas las características industriales de la CIAA.

FIGURA 1.2: Kit EDU-CIAA-NXP.

La EDU-CIAA-NXP fue concebida como una introducción económica a la programación de la CIAA-NXP y ambas comparten el mismo modelo y marca de microcontrolador. Se fabricaron miles de unidades.

Las características básicas son las siguientes:

- Dos puertos USB.
- Tres luces led. Una luz multicolor.
- Cuatro pulsadores.
- Puerto de comunicación serie con bornera.
- Dos conectores de expansión.

La placa no cuenta con salida video, audio o conectividad inalámbrica. Sin embargo existe la posibilidad de expandir sus capacidades originales mediante el diseño de otra placa.

1.2. Descripción del proyecto

Se desarrolló e implementó una placa de expansión y firmware que agrega las siguientes capacidades multimedia a la EDU-CIAA-NXP:

- Salida de video compatible con televisores de alta definición.
- Audio estéreo.

[1]Programa informático que define el funcionamiento básico de un dispositivo electrónico programable.

[2]Un dato curioso: en el año 2015, la empresa NXP compró a su competidor Freescale, fusionando su catalogo de productos.

- Conectividad inalámbrica (Wi-Fi / Bluetooth).
- Dos entradas de Joystick.
- Memoria no volátil.

En la figura 1.3 se observa la placa base EDU-CIAA-NXP y la placa de expansión multimedia. De igual forma se evidencian las nuevas capacidades que la expansión ofrece.

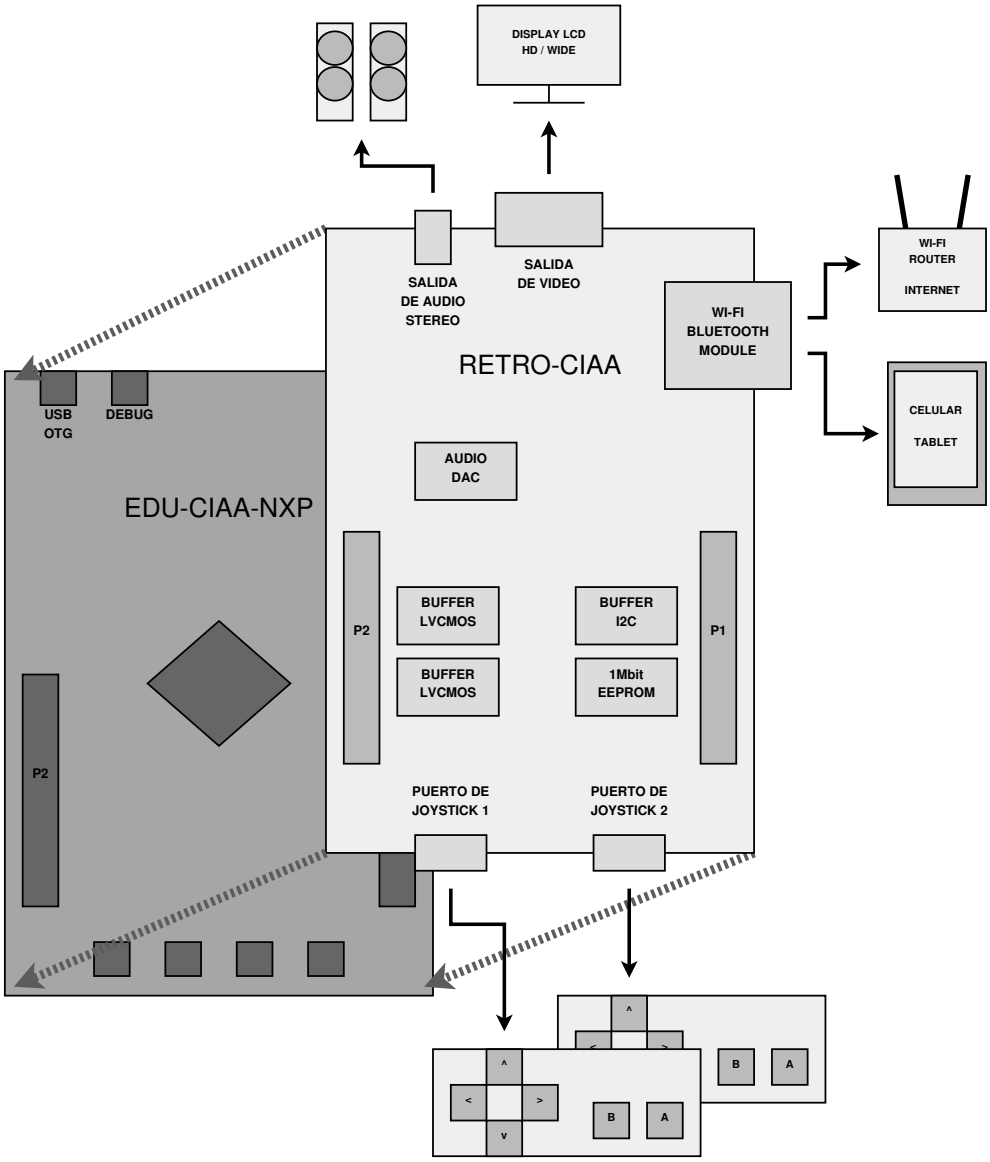


FIGURA 1.3: Expansión multimedia sobre el kit EDU-CIAA-NXP.

La EDU-CIAA-NXP no ha sido diseñada para aplicaciones multimedia y los recursos disponibles son limitados. En consecuencia, la resolución de imagen y la cantidad de colores disponibles es relativamente baja para los estándares actuales.

A fin de transformar estas limitaciones en un atractivo y considerando que los destinatarios de este producto serán tanto adolescentes como jóvenes adultos

que crecieron con el *Family Game*[3] u otras consolas de videojuegos similares, es que esta expansión incorporó dos puertos de joystick para mandos estándar de bajo costo y conceptualmente, tomó la forma de una consola *retro*[4] que comparte aquellas características limitadas.

De ahí surgió el nombre de esta expansión multimedia: RETRO-CIAA.

1.3. Motivación

La motivación nació como consecuencia de las siguientes observaciones:

- Las interfaces gráficas de usuario son cada vez más comunes en todo tipo de sistema embebido de última generación en automóviles, electrodomésticos, alarmas, herramientas de todo tipo, etc.
- Los usuarios están habituados a interactuar diariamente con interfaces gráficas, siendo cada vez más renuentes a interfaces tradicionales construidas con indicadores luminosos, displays de ocho segmentos, etc. y que resultan anti-intuitivas o comercialmente poco atractivas para el estándar actual.
- Por cuestiones de costo, capacidad, consumo u otra necesidad especifica, no siempre es viable incorporar un ecosistema de interfaz gráfica ya resuelto corriendo sobre un sistema operativo convencional.

El conocimiento en *Computación Gráfica* permite:

- Hacer un uso óptimo de la funcionalidad gráfica de un hardware determinado, utilizando menos recursos para cumplir con los mismos requerimientos o bien, mejorando las prestaciones con los mismos recursos.
- Reducir costos y consumo de energía.
- Implementar video de alta performance en sistemas que no soportan esa funcionalidad.
- Desarrollar nuevas interfaces gráficas y algoritmos.

Sin embargo, esa disciplina es muy poco común en nuestro país.

- No se dicta en la currícula de las diferentes ingenierías, en terciarios, secundarios, o carreras de especialización.
- Actualmente no se ofrece ningún curso introductorio.
- No existe una herramienta para estudiar estos principios vinculados directamente a los sistemas embebidos o en un muy bajo nivel de abstracción.

En resumen, el valor de este trabajo radica en el intento de promover un área de gran importancia para los sistemas embebidos pero actualmente no difundida en el país.

---

[3] Clon de la consola Nintendo Famicom, muy popular en Latinoamérica en los años '90.

[4] Dicese de algo que imita el estilo o la moda del pasado reciente.

Computación Gráfica

Es la generación o procesamiento de imágenes por computadora. Es una especialidad que está íntimamente ligada a la aparición y desarrollo de los videojuegos, los efectos visuales por computadora en películas, las interfaces gráficas de usuario, etc. Se vincula ampliamente con nuevos campos como la «Visión por Computadora» utilizada en reconocimiento facial, vehículos autónomos, procesos industriales automatizados, etc.

Por eso, la forma de consola de videojuegos no es casual en una herramienta pedagógica destinada a la practica de esta disciplina.

1.4. Objetivos y alcance

El proyecto RETRO-CIAA aspira a ser:

1. Un *framework* (conjunto coherente de funciones y utilidades) simple para implementar videojuegos o algoritmos de computación gráfica.
2. Una herramienta didáctica de bajo nivel de abstracción para comprender en detalle el funcionamiento de un controlador de video, su driver y las funciones gráficas para dibujar en pantalla.

El presente trabajo pretende ser una herramienta útil para la enseñanza o práctica de computación gráfica en un entorno de muy bajo nivel de abstracción. Se espera que la comunidad educativa pueda sumarse a la propuesta brindando cursos o talleres sobre computación gráfica utilizando esta nueva herramienta que, por estar basada en la EDU-CIAA-NXP, en gran medida ya les es familiar.

Para lograr estos objetivos, se evaluó que la expansión debía poseer las siguientes características:

1. Diseño de hardware y software simple, de fácil comprensión
2. Muy alta usabilidad del firmware.
3. Excelente performance de audio y video.
4. Muy bajo costo.

A quien va dirigido

La expansión se orienta a los actuales poseedores de una EDU-CIAA-NXP y también a los hobbistas de la electrónica o entusiastas de la programación de videojuegos debido a su atractivo como consola de videojuegos retro y de código y diseño abierto, cubriendo así un amplio abanico de posibles usuarios, promoviendo el uso de la EDU-CIAA-NXP y extendiendo el alcance del proyecto por fuera de un posible uso en el ámbito educativo formal.

En los últimos 20 años ha ocurrido un cambio de paradigma como consecuencia de los avances constantes en la capacidad de los microcontroladores, redes de comunicación y sistemas digitales. Los ingenieros electrónicos están adentrándose (algunos por primera vez) en el mundo de la programación en lenguaje 'C'. Del

mismo modo, este trabajo espera motivar a programadores de carrera a transitar el camino contrario: comenzar a adentrarse en el mundo de la electrónica, los sistemas embebidos y la programación de bajo nivel a través de una actividad divertida como es el desarrollo de videojuegos.

Alcance

El trabajo incluyó las siguientes tareas [6]:

- Evaluación de las capacidades de la EDU-CIAA-NXP con el objetivo de determinar la factibilidad técnica y el mejor uso posible de recursos por parte de la expansión multimedia.
- Identificación del modo de video y formato de sonido más óptimo; con menor uso de recursos y que a la vez sea aceptable para cumplir los objetivos del proyecto.
- Pruebas de concepto; programación del núcleo Cortex-M0 y generación de señales de sincronismo de video y pixeles a través de un DAC ad-hoc de muy bajo costo.
- Identificación de nuevos periféricos que puedan ser de utilidad.
- Evaluación de la factibilidad económica del conjunto.
- Desarrollo del diagrama esquemático y lista de materiales.
- Desarrollo de la placa base.
- Desarrollo de dos versiones: una muy económica (lite) y otra completa (full), diferenciándose las dos solo por los componentes opcionales montados en la misma placa base.
- Montaje de al menos cinco prototipos en versiones full y lite para realizar pruebas funcionales y desarrollo de aplicaciones.
- Desarrollo del firmware y ejemplos básicos de uso.
- Documentación completa.
- Diseño de página web del proyecto con tutoriales, bibliografía y otros recursos orientados al uso y programación de la expansión multimedia.

El trabajo no incluyó:

- Simulador multiplataforma para desarrollo y ejecución de las aplicaciones en una PC. Será motivo de otro proyecto.
- Simulador corriendo en navegador web. Será motivo de otro proyecto. Firmware que implemente un sistema operativo y capacidad de grabar en flash múltiples aplicaciones y ejecutar alguna en particular. Será motivo de otro proyecto.
- Producción en masa, packaging, etc. Si es comercialmente viable, puede encararse como otro proyecto.
- Certificaciones de ningún tipo.
- Todo lo que esté por fuera del alcance definido.

Capítulo 2

Introducción Específica

2.1. Descripción técnica

En la presente sección se exponen precisiones técnicas generales.

2.1.1. Plataforma

Se decidió que el proyecto se materialice como una expansión de la EDU-CIAA-NXP en base a las siguientes consideraciones:

- El uso de la EDU-CIAA-NXP como herramienta educativa en secundarios, universidades y cursos de especialización, habiéndose entregado a la fecha un aproximado de 2500 unidades.
- El microcontrolador de la EDU-CIAA-NXP (LPC4337JBD144 de la empresa NXP) cuenta con dos núcleos ARM Cortex-M; un Cortex-M4 y un Cortex-M0. Generalmente se utiliza el núcleo M4 para correr el programa principal mientras se desaprovecha -incluso relegando al olvido- al segundo.
- Se comprobó que es factible implementar un controlador de video, rutinas de audio y conectividad en el núcleo M0, con cero impacto en las aplicaciones corriendo en el núcleo M4 habitual.
- Una expansión de bajo costo por encima de una plataforma ya conocida, de amplia disponibilidad y sobre la cual ya se trabaja puede acelerar el desarrollo, adopción y uso de la expansión

2.1.2. Hardware

Consiste en una expansión («poncho» en la terminología del proyecto CIAA) implementada en un PCB de dos capas. Se conecta sobre la EDU-CIAA-NXP a través de sus dos tiras de pines denominadas P1 y P2.

La expansión esta conformada por:

- Buffers para las entradas y salidas digitales.
- Circuito de resistencias para generar una señal de video analógica a partir de salidas digitales.
- DAC de audio estéreo por protocolo I^2S.

- EEPROM I^2C para almacenamiento no volátil.
- Modulo Wi-Fi/Bluetooth con antena integrada e interfaz serie.
- Conversor de tensiones I^2C para comunicación por protocolo E-DDC a través del conector de video.

2.1.3. Firmware

Se hace un uso intensivo de los dos núcleos asimétricos presentes en el microcontrolador de la placa EDU-CIAA-NXP.

En el núcleo Cortex-M0 se ejecuta el controlador de video por software y en el núcleo Cortex-M4 se ejecuta la aplicación que demanda poder de procesamiento. Los dos núcleos están constantemente intercomunicados.

2.1.4. Características

En la tabla 2.1 se enumeran las características del producto que resulta de expandir la EDU-CIAA-NXP con un poncho RETRO-CIAA y de utilizar el nuevo firmware desarrollado para este proyecto.

TABLA 2.1: Características principales de EDU-CIAA-NXP + RETRO-CIAA

Parámetro	Valor
Velocidad de reloj	204 MHz
Memoria del sistema	64 KiB disponible para firmware y aplicaciones
Memoria de video	72 KiB front/back buffers
Señal de video	1280x720, 60 Hz (HDTV 720p)
Resolución lógica de video	256x144
Formato de pixel	RGB 3:3:2, 1 byte por pixel
Colores	256
Salida de video	DB15HD Hembra, Analógico, pinout VGA
Entrada de Joysticks	2 x DB9 Macho, compatibles con Family Game
Formato de Audio	16 bit, 32 kHz, estéreo
Salida de Audio	2,5 mm audio jack, estéreo
Memoria no volatil	EEPROM I^2C, 1 Mibit

2.2. Requerimientos

En esta sección se comentará acerca de los requerimientos [6] y sus modificaciones a lo largo del desarrollo del proyecto [4].

Si corresponde, se realizarán aclaraciones en cursiva al final de cada punto.

2.2.1. Requerimientos asociados al costo

(R-1.1) El costo total de materiales de la versión full no debe superar el $50\,\%$ del costo de venta al público de un kit EDU-CIAA-NXP.

2.2.2. Requerimientos asociados al PCB

(R-2.1) Debe ser de material FR4 doble faz.

(R-2.2) Debe encastrar en los puertos P1 y P2 de la EDU-CIAA-NXP.

(R-2.3) No debe bloquear el acceso a los pulsadores y periféricos existentes en el kit EDU-CIAA-NXP.

(R-2.4) Existen sutiles diferencias entre versiones del kit EDU-CIAA-NXP. El proyecto debe ser compatible con «v0.2», «v1.1» y «v1.2».

2.2.3. Requerimientos asociados a la electrónica

(R-3.1) Se utilizarán buffers entre entradas y salidas a la EDU-CIAA-NXP.

(R-3.2) Uso de capacitores para filtrar ruido de alta frecuencia (bypass capacitors) y complementar bajas de tensión por variaciones momentáneas de consumo.

(R-3.3) En la medida que sea posible los componentes serán de tecnología SMD.

(R-3.4) Los componentes SMD serán de tamaño mínimo 1608 métrico (0603) para pasivos y de paso $0{,}65\,\mathrm{mm}$ mínimo entre pines para encapsulado de circuitos integrados.

2.2.4. Requerimientos asociados con la salida de video

(R-4.1) La señal de video corresponde a un modo de 1280x720 píxeles a 60 Hz. *El modo 720p con relación de aspecto 16:9 entre ancho y alto es el estándar mínimo implementado en los televisores y monitores modernos.*

(R-4.2) La señal de video generada será análoga, de 0 a 0,7 V para los componentes de color rojo, verde y azul a $75\,\Omega$ de impedancia y de $0\,\mathrm{V}$ a $3{,}3\,\mathrm{V}$ (LVCMOS) para el sincronismo vertical y horizontal. *Tipo de conexión económica y simple de implementar con resistencias. El costo de incluir un integrado transceiver HDMI encarece el proyecto en una característica que no es opcional.*

(R-4.3) El sincronismo vertical y horizontal serán positivos; el flanco ascendente marca el inicio del sincronismo y el descendente el fin.

(R-4.4) La señal generada deberá tener una exactitud de $100\,\mathrm{ns}$ o menos.

(R-4.5) La conexión física será idéntica al estándar VGA: conector D-Sub HD15 hembra y con el mismo pinout.

(R-4.6) El tamaño de cada pixel en memoria será de un solo byte (8 bits), generando un total de 256 colores posibles. *La memoria del microcontrolador es escasa (136 kB). No es posible incluir memoria externa. Un tamaño menor al byte*

enlentece el procesamiento de pixeles y los gráficos serían demasiado simples aun para una consola de 8 bits.

(R-4.7) Se describe el color de cada pixel dentro el mismo byte del pixel utilizando tres bits para indicar valor del componente Rojo, tres para Verde y dos para Azul (RGB 3:3:2). *Cada bit de pixel se mapea a una salida del microcontrolador y a una entrada del DAC ad-hoc que genera la señal de video. Se asignan dos bits al componente azul debido a la menor sensibilidad del ojo humano a ese componente.*

(R-4.8) La salida de un pixel por los puertos del microcontrolador debe coincidir con la representación en memoria del pixel. *Respetar formato de bits RGB 3:3:2. Son ocho salidas: tres para el componente rojo, tres para verde y dos para azul.*

(R-4.9) Las ocho salidas de pixel del microcontrolador deberán pertenecer al mismo puerto y en lo posible ser bits contiguos. *Necesario para establecer el estado de todo el puerto en una sola operación de escritura.*

(R-4.10) Se implementará un «framebuffer» o área de memoria que contiene todos los pixeles que se muestran en pantalla. *Necesario para implementar algoritmos de computación gráfica como dibujo de líneas.*

(R-4.11) El framebuffer debe residir en un área de memoria contigua. *Para dibujar se accede directo a los pixeles asumiendo que es un array.*

(R-4.12) La resolución lógica será de 256x144, cinco veces menor al modo de video utilizado. El costo total en memoria del framebuffer (Memoria de video) será de 36 kB. *En el microcontrolador LPC-4337 la memoria de 136 kB está repartida en 4 bancos. Los dos bancos RamLoc de 32 y 40 kB se encuentran en áreas no contiguas. Los bancos RamAHB de 32, 16 y 16 kB se encuentran en áreas contiguas sumando 64 kB. Por motivos de rendimiento, el framebuffer debe ser igual o múltiplo de la resolución de la señal de video. La división por cuatro (320x180) consume 56,25 kB, la división por tres no es un número entero y por dos supera la cantidad de memoria total.*

2.2.5. Modificaciones

En el informe de avance al 29/8/2018 se hace mención de dos requerimientos que no se cumplirán [4]. El error en ambos fue que ninguno debió haberse especificado como un requerimiento en primer lugar:

Requerimiento R-4.3, modos de sincronismo

Este requerimiento depende del modo de sincronismo implementado. No todos los modos de sincronismo se especifican por sincronismo positivo. Experimentando con los prototipos se comprobó que usar CVT (Coordinated Video Timings) funciona mejor que simular un modo HD de televisión de sincronismo positivo como estaba previsto.

Requerimiento R-4.12, tamaño de framebuffer

Implementar un esquema de doble buffer («ping-pong») mejora sustancialmente la performance del sistema y la estabilidad de la imagen. Por ese motivo, actualmente el costo total de memoria no es uno sino dos buffers de 36 kB (72 Kb total).

2.2.6. Otros requerimientos

Desde el inicio, el objetivo fue lograr alta performance en el sistema de audio y video. Esta importante premisa que guió el diseño de hardware y software, fue omitida en todos los documentos previos. En esta memoria se documenta exhaustivamente al respecto en las decisiones de diseño y se refleja en los ensayos y resultados.

2.3. Tareas

Se dividió el trabajo en las siguientes tareas [6]:

- Búsqueda de antecedentes de trabajos similares. 10 h.
- Evaluación de factibilidad técnica y pruebas de concepto. 35 h.
- Definición de componentes y circuitos integrados necesarios. 30 h.
- Diseño del diagrama esquemático. 50 h.
- Diseño del PCB. 90 h.
- Armado de 5 prototipos a mano. 10 h.
- Pruebas funcionales, drivers básicos para integrados. 30 h.
- Firmware base y utilidades. 40 h.
- Firmware gráfico. 100 h.
- Firmware para audio. 70 h.
- Herramientas y utilidades. 30 h.
- Aplicaciones de ejemplo. 80 h.
- Redacción de tutoriales sobre el uso del producto. 40 h.
- Documentación completa. 40 h.
- Armado de página web del proyecto. 40 h.

2.4. Planificación

En la figura 2.1 se presenta el diagrama Activity On Node. La planificación contempla un total de 695 horas de trabajo necesarias para finalizar las tareas [6].

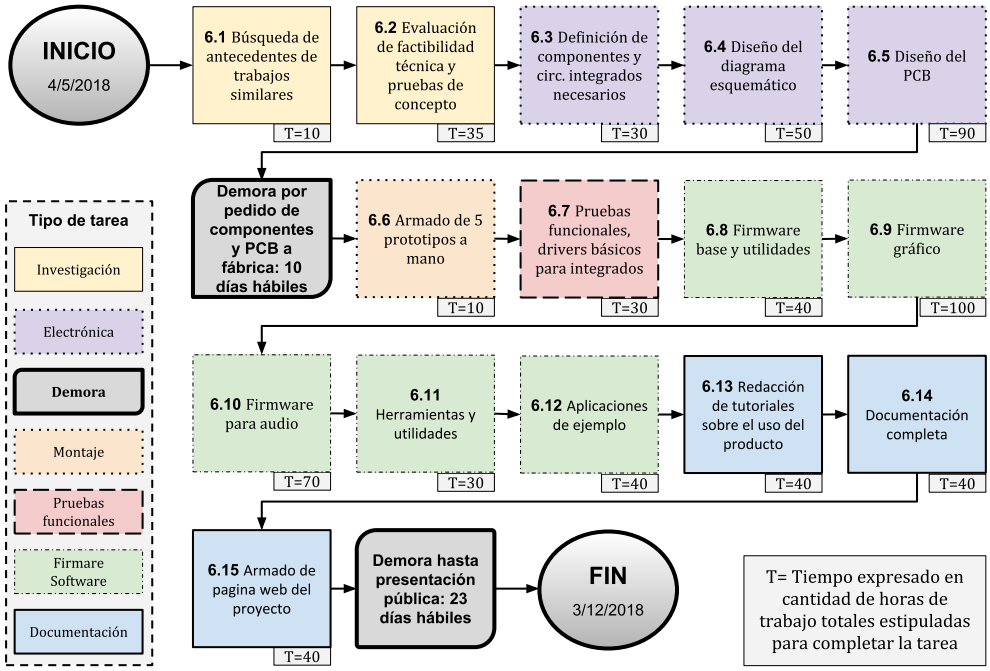


FIGURA 2.1: Diagrama Activity On Node [6, p.14]. Las tareas se planifican de forma secuencial debido a que solo habrá una persona trabajando en ellas.

2.5. Avances

2.5.1. Documentación existente

El estado de avance del trabajo se documentó en el *Informe de Avance al 29/8/2018* que fue un documento presentado en la cursada del CESE [4] y el *Informe de Avance al 28/10/2018* que es un documento interno [5].

2.5.2. Diagrama de Gantt

En la figura 2.2 se presenta el diagrama de Gantt con el estado de avance al 29/8/2018. A esa fecha se avanzó según la planificación prevista.

Tarea	Tiempo
6.1 Búsqueda de antecedentes de trabajos similares	1d 4h
6.2 Evaluación de factibilidad técnica y pruebas de concepto	5d 5h
6.3 Definición de componentes y circ. integrados necesarios	5d
6.4 Diseño del diagrama esquemático	8d 2h
6.5 Diseño del PCB	15d
Demora por pedido de componentes y PCB a fábrica	13d 2h
6.6 Armado de 5 prototipos a mano	1d 4h
6.7 Pruebas funcionales, drivers basicos para integrados	5d
6.8 Firmware base y utilidades	6d 4h
6.9 Firmware gráfico	16d 4h
6.10 Firmware para audio	11d 4h
6.11 Herramientas y utilidades	5d
6.12 Aplicaciones de ejemplo	6d 4h
6.13 Redacción de tutoriales sobre el uso del producto	6d 4h
6.14 Doc completa en formato LATEX	6d 4h
6.15 Armado de pagina web del proyecto	6d 4h
Demora hasta presentacion pública	23d

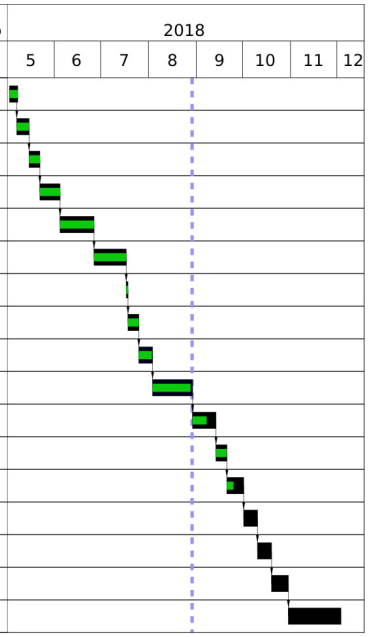


FIGURA 2.2: Diagrama de Gantt. Informe de avance al 29/8/2018.

Capítulo 3

Diseño

En el presente capitulo se revisarán las decisiones de diseño tomadas para cumplir con los requerimientos del proyecto.

3.1. Introducción

El diseño se dividirá en secciones que se corresponden con las nuevas características que agrega la expansión o con un área de trabajo determinada:

- Video
- Audio
- Entradas de joystick
- Wi-Fi / Bluetooth
- Almacenamiento
- Buffers de E/S
- Esquemático
- PCB
- Firmware

3.2. Video

Este subsistema es el mas complejo y sobre el cual recae la mayor parte de los requerimientos. El tema se dividirá en diferentes áreas de trabajo correlativas para lograr un claro abordaje de todos los conceptos. Cada área presentará los principios necesarios para avanzar a la siguiente.

3.2.1. Consideraciones

Se prestó especial atención en explotar al máximo las capacidades del microcontrolador LPC4337JBD144 presente en la EDU-CIAA-NXP.

Este microcontrolador de NXP dispone de dos núcleos asimétricos[1]. El diseño asimétrico responde a una aplicación de uso común en sistemas embebidos en donde el núcleo Cortex-M0 (simple, de bajo consumo) recibe y procesa eventos por interrupción[2]. Al acumular suficientes datos, despierta al núcleo Cortex-M4 (potente, de mayor consumo) quien rápidamente realiza el procesamiento pesado aprovechando sus instrucciones DSP y SIMD. Al terminar la tarea, avisa al núcleo Cortex-M0 y vuelve al estado de bajo consumo.

En RETRO-CIAA, las tareas asignadas a cada núcleo son las siguientes:

Cortex-M0 Se utiliza como un coprocesador. Genera las señales eléctricas que componen la salida de video.

Cortex-M4 Corre la aplicación principal que hará uso de funciones del firmware para generar diferentes tipos de gráficos en tiempo real.

En cuanto al uso y consumo:

Cortex-M0 La mayor parte del tiempo corre a la máxima velocidad permitida en la hoja de datos.

Cortex-M4 La intensidad de uso depende exclusivamente de la aplicación.

En ambos núcleos se utiliza la máxima frecuencia posible de reloj, la cual se especifica en la ecuación 3.1 y da como resultado el periodo de cada ciclo en la ecuación 3.2. Este periodo es generalmente idéntico al tiempo de ejecución de una instrucción de un solo ciclo, aunque no se puede tener una certeza absoluta ya que los núcleos M0 y M4 son procesadores con un pipeline de 3 etapas y la tarea de ejecución se procesa en paralelo junto a la de obtención y decodificación de otras instrucciones [26] [27].

$$MCU_{clock} = 204\,\mathrm{MHz} \tag{3.1}$$

$$MCU_{cycle} = \frac{1}{MCU_{clock}} = \frac{1}{204\,\mathrm{MHz}} = 4{,}901\,960\,784\,31\,\mathrm{ns} \tag{3.2}$$

A la máxima velocidad de reloj ambos núcleos son suficientemente potentes para las tareas asignadas, logrando una excelente performance de audio y video.

3.2.2. Restricciones

Se enumeran restricciones encontradas al encarar el diseño del subsistema de video.

- La memoria total disponible. Dividida en diferentes bancos de SRAM y accesible por ambos núcleos, es de solo 136 KiB.
- Los pines disponibles del microcontrolador. No es posible usar 8 bits correlativos de un mismo puerto.

[1] Diferente set de instrucciones, prestaciones y consumo ya que atienden diferentes tareas especificas.

[2] Por ejemplo, la lectura de un sensor a intervalos regulares.

- La disposición de las pistas y el diseño no ideal del plano de tierra en el PCB de la EDU-CIAA-NXP dificulta enormemente trabajar con señales de alta velocidad.
- El regulador de $3{,}3\,\mathrm{V}$ de la EDU-CIAA-NXP no es de bajo nivel de ruido y solo entrega máximo $1\,\mathrm{A}$, bajando considerablemente la tensión cuando se llega a los $800\,\mathrm{mA}$ de consumo.

3.2.3. Conceptos básicos

A continuación se presentan conceptos básicos de computación gráfica necesarios para avanzar en la lectura de esta memoria:

Pixel La mínima unidad que forma una imagen digital. El pixel guarda el color en un punto de la imagen.

Componentes de color Composición del color según la intensidad individual de los tres primarios que sumados lo conforman: rojo, verde y azul. Se expresa como RGB (Red, Green, Blue).

Formato de pixel Tamaño (8, 16, 24, 32 bits, etc) e interpretación de los bits en cada pixel para manipular su color. A mayor cantidad de bits, mayor cantidad de colores posibles.

Resolución Cantidad de pixeles en el área visible de una pantalla. Al ser un dispositivo bidimensional, generalmente se indica como cantidad de *pixeles en ancho* x *pixeles en alto*. Por ejemplo, 1024x768.

Relación de aspecto Es la proporción entre el ancho y el alto de una imagen. Normalmente se expresa como *ancho* : *alto*. Por ejemplo, 4:3.

Framebuffer Memoria contigua en RAM que contiene todos los pixeles que forman la imagen que se verá en pantalla.

Doublebuffer Se definen dos Framebuffer de igual tamaño. Uno asume el rol de front y el otro de back. Mientras la aplicación dibuja en back, el adaptador de video genera la señal a partir de front. Al terminar front, se intercambian los roles (front por back, back por front) y se empieza una nueva pantalla.

Pixmap Conjunto de pixeles de un ancho, alto y formato de pixel determinado. Generalmente son imágenes (dibujos, patrones, fotos, etc) guardadas en memoria no volátil y eventualmente copiadas al Framebuffer para su visualización.

Adaptador de video Dispositivo electrónico que se ocupa de convertir el contenido del Framebuffer en una señal de video que sera recibida, decodificada y visualizada en una pantalla.

Frecuencia de refresco Cada cuantos ciclos por segundo (Hz) se re-dibuja una pantalla completa. Si la frecuencia no es lo suficientemente rápida, la ilusión de movimiento se pierde.[3]

Modo de video La resolución (relación de aspecto), formato de pixel y frecuencia de refresco elegidos para generar imágenes desde un adaptador de video.

[3] Ver «Fenómeno phi». https://es.wikipedia.org/wiki/Fen%C3%B3meno_phi

Vertical Blanking Period Lapso de tiempo en el cual el adaptador de video no enviá datos visibles a la pantalla. Comienza luego de procesar la ultima linea de la pantalla actual y termina apenas antes de procesar la primer linea de la siguiente pantalla.

Vertical Blanking Interrupt Interrupción que se dispara al iniciarse el periodo de Vertical Blanking.

VGA «Video Graphics Array» es un estándar de video analógico desarrollado en los años '80 para la IBM PC. Puede referirse tanto a las distintas señales de video analógico que lo conforman, al conector DB15-HD que las agrupa o al modo de video de 640x480 pixeles característico del estándar.

Video compuesto Antigua señal de video analógico codificada de modo de transmitirse por un único cable coaxial de $75\,\Omega$ de impedancia. En dispositivos de consumo masivo el conector es «RCA» de color amarillo. Los estándares de codificación mas utilizados son «PAL» o «NTSC», con diferentes variantes propias del país que los adopta («PAL-B», «PAL-N», «PAL-M», etc).

3.2.4. Trabajos similares

Se revisarán cuatro proyectos open source que generan señales de video desde un microcontrolador. El objetivo es conocer el estado del arte y poner en contexto las decisiones de diseño propias de este trabajo. Se describirán las técnicas utilizadas resaltando ventajas y desventajas.

XGameStation PIC 16-Bit

Consola de Videojuegos educativa. Proyecto comercial desarrollado por André LaMothe, reconocido autor de libros sobre desarrollo de videojuegos [14]. Microcontrolador PIC24HJ256. 16 KiB RAM, 256 KiB Flash, 80 MHz, 40 MIPs. Abreviado como «XGS».

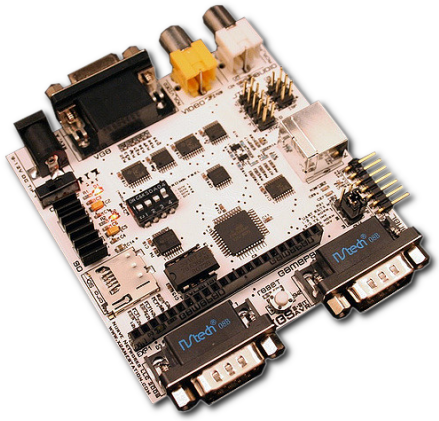


FIGURA 3.1: XGameStation PIC 16-Bit.

HYDRA Game Development Kit

Consola de Videojuegos educativa, tambien de André LaMothe [15]. Microcontrolador Parallax Propeller. 32 KiB RAM, 128 KiB EEPROM, 80 MHz, 160 MIPs. Abreviado como «HYDRA».

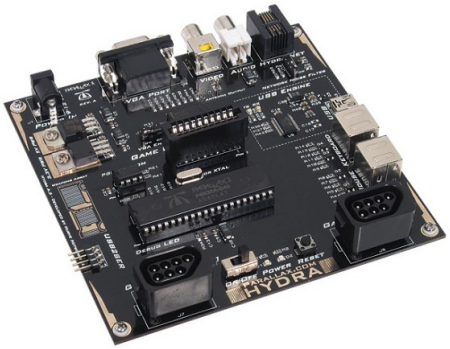


FIGURA 3.2: Hydra Game Development Kit.

The Uzebox Project

Consola de Videojuegos, proyecto open source iniciado por Alec Bourque [1]. Microcontrolador ATmega644. 4 KiB RAM, 64 KiB Flash, 28,618 18 MHz. Abreviado como «UZEBOX».

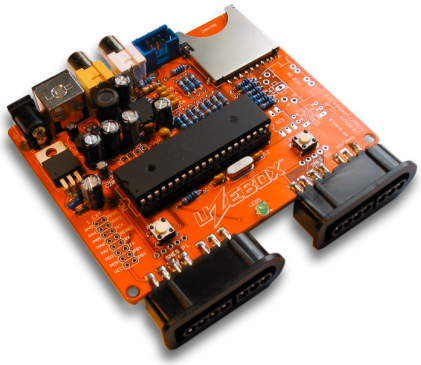


FIGURA 3.3: Uzebox EX1.

Speecy AMP

Port a EDU-CIAA-NXP de emulador de ZX Spectrum[4], proyecto personal de Telmo Moya [17]. Microcontrolador LPC4337. 136 KiB RAM, 1 MiB Flash, 204 MHz. Abreviado como «SPEECY».

[4]Computadora personal de principios de los años '80.

Caracteristicas generales

Los microcontroladores son bastante heterogéneos entre si. Dos de ellos son multinúcleo pero de diferente tipo:

«HYDRA» es un sistema SMP o *Symmetric Multiprocessing*: dispone de 8 núcleos idénticos, cualquiera de ellos puede ocuparse de la misma tarea.

«SPEECY» es un sistema AMP o *Assymetric Multiprocessing*: los 2 núcleos tienen distintas prestaciones y/o acceso a recursos según una tarea típica asignada en tiempo de diseño.

En cuanto a la conectividad de video, todos disponen de salida VGA y todos excepto «SPEECY» disponen de salida de video compuesto.

Diseño del sistema de video

La resolución de pantalla típica utilizada es 256x192 pixeles en video compuesto o 640x480 en VGA, siempre en relación de aspecto 4:3.

El formato de pixel y la cantidad de colores efectivos esta limitado por la electrónica implementada en la etapa de salida de video:

- («XGS», «HYDRA») 2 bits por componente de color (se indica como RGB 2:2:2).
- («UZEBOX») 3 bits en componentes Rojo y Verde, 2 bits en componente Azul (RGB 3:3:2).
- («SPEECY») 1 bit por componente (RGB 1:1:1).

Las restricciones de memoria para implementar un Framebuffer han sido salvadas de la siguiente forma:

- («HYDRA») Reduciendo la cantidad de colores por pixel, utilizando 4 bits (2^4, 16 colores) o menos.
- («SPEECY») Reduciendo el área activa de la pantalla. En un modo de video de 640x480, solo se actualizan los pixeles en un sector de 192x256.
- («XGS», «UZEBOX») No implementando Framebuffer[5].

Los proyectos que implementan un solo Framebuffer («SPEECY», «HYDRA») debieron decidirse por una de dos opciones no ideales:

- («SPEECY») Para no sacrificar performance, se escribe sobre la misma área en memoria que se esta leyendo para generar la señal de video, creando anomalías y parpadeos visibles.
- («HYDRA») Para evitar anomalías visuales, se realiza el dibujo de la nueva pantalla solo en el periodo de Vertical Blanking.

«SPEECY» genera señal de video VGA desde el núcleo M0 utilizando DMA para cambiar el estado de 3 pines que conforman un bit por cada componente de color, con un máximo de 8 colores (2^3). Como cada pixel ocupa 8 bits en memoria,

[5]Implementar un Framebuffer es un requerimiento inamovible del proyecto RETRO-CIAA. La discusión de técnicas alternativas excede el alcance de esta memoria.

5 bits son desaprovechados. La señal de video producida es VGA en 640x480 pixeles. El Framebuffer tiene un tamaño de $48\,\mathrm{KiB}$, mucho menor al necesario para soportar 640x480 en resolución completa. La solución implementada fue actualizar un área de pantalla de solo 192x256 pixeles, $16\,\%$ del tamaño de pantalla. No existe sincronización entre el núcleo M0 y el M4, ambos funcionan de manera independiente [17].

«HYDRA» debe dibujar la pantalla en un tiempo extremadamente acotado (el que dure el Vertical Blanking Period) lo que limita la capacidad de dibujar gráficos complejos. No obstante tiene un gran poder de procesamiento simétrico disponible para la aplicación [13].

Los sistemas de un solo núcleo («UZEBOX», «XGS») ocupan mas del $90\,\%$ de su tiempo en generar la señal de video con los pixeles visibles de la pantalla actual, dejando muy poco tiempo de procesador para la aplicación; solo corre en el periodo de Vertical Blanking [2] [9].

Conclusiones

La revisión de estos proyectos generó como resultado un conjunto de características deseables:

La señal de video compuesto es obsoleta. Generar color en «PAL» o «NTSC» es complejo y la calidad de imagen es mala. Se decide no soportarla.

Los monitores CRT[6]y las resoluciones 4:3 son obsoletas. En su lugar se decide soportar el estándar actual de HDTV (Television Digital de Alta Definicion) diseñando con las ventajas y limitaciones de las pantallas LCD en mente.

Del mismo modo se abandona la relación de aspecto 4:3 para brindar soporte a 16:9 de HDTV, lo que anteriormente era llamado «widescreen» para diferenciarlo del antiguo estándar.

En cuanto a la implementación del Framebuffer, no es aceptable sufrir anomalías en la imagen en pantalla o limitar la complejidad gráfica por el reducido tiempo en el cual se debe dibujar. Se decide utilizar Doublebuffer aunque ello implique sacrificar resolución para compensar el uso de memoria.

Finalmente, se decide implementar formato de pixel en un byte y aprovechar completamente sus bits para soportar $2^8 = 256$ colores distintos.

3.2.5. Adaptador de video externo

La posibilidad de utilizar un adaptador de video externo también fue evaluada. Estos dispositivos implementan las funciones primitivas de dibujo, el framebuffer y el manejo de la señal de video. Se comunican con el el microcontrolador a través de un puerto de comunicación y mediante un protocolo reciben comandos de dibujo para colocar pixeles y copiar o borrar áreas en su memoria de video.

Los dispositivos evaluados fueron los siguientes:

[6] «Catodic Ray Tube» Tubo de Rayos Catódicos fue una tecnología que dominó los televisores y monitores hasta bien entrado el siglo XXI.

DisplayLink por USB

El dispositivo se conecta al televisor por HDMI y recibe comandos a través del puerto USB-OTG del host. La implementación del driver USB y el port de la librería de comandos a la EDU-CIAA-NXP se realizó como trabajo final del CESE. En la memoria del trabajo se indica que el costo del dispositivo es de 70 USD [7]. En el video de referencia [8] se puede observar funcionamiento y performance.

Controlador ILI9341

Es posible utilizar un controlador compatible con el ILI9341 para manejar una pantalla LCD o incluso se podría generar una señal VGA. El proyecto SPEECY revisado en la subsección 3.2.4 tiene otra versión corriendo en un LCD con ILI9341 [17] y también hay un video disponible [18] para evaluar su funcionamiento.

Conclusiones

Este trabajo apunta a correr aplicaciones que requieren una extraordinaria fluidez y velocidad de dibujo para implementar videojuegos o algoritmos de computación gráfica en tiempo real.

El método de preparar un mensaje, enviarlo, esperar a que el adaptador de video reciba, interprete y procese, claramente no tiene la misma inmediatez ni determinismo que acceder al Framebuffer mediante un simple acceso a memoria local.

En consecuencia, con los dispositivos revisados no es posible obtener ni la performance ni el costo al que apunta este trabajo.

3.2.6. Formato de pixel

Cada pixel ocupa 1 byte (8 bit). Los componentes de color de cada pixel se asignan de forma directa (en el mismo pixel) desde el bit mas al menos significativo (MSB a LSB) de este modo: rojo 3 bit, verde 3 bit, azul 2 bit. Este formato es abreviado por sus siglas en ingles como RGB 3:3:2. En la figura 3.4 se ve una representación del Framebuffer y el formato de los pixeles en cada byte.

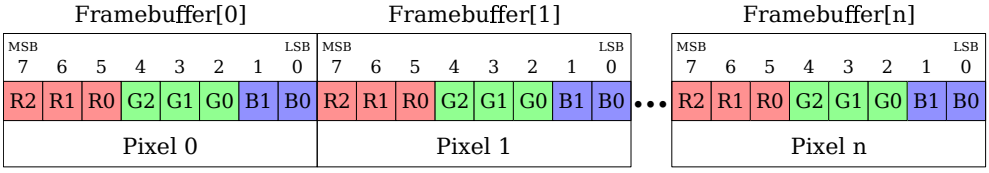


FIGURA 3.4: Formato de pixeles. Nótese que cada uno de los 'n' bytes del Framebuffer es un pixel. Y cada pixel de 8 bit esta conformado por 3 bit para rojo (R2-R0), 3 bit para verde (G2-G0) y 2 bit para azul (B1-B0).

Como se muestra en la figura 3.5, la asignación de los 2 bit al componente azul se debe a la menor sensibilidad del ojo humano (conos tipo S) a ese rango de longitud de onda del espectro visible.

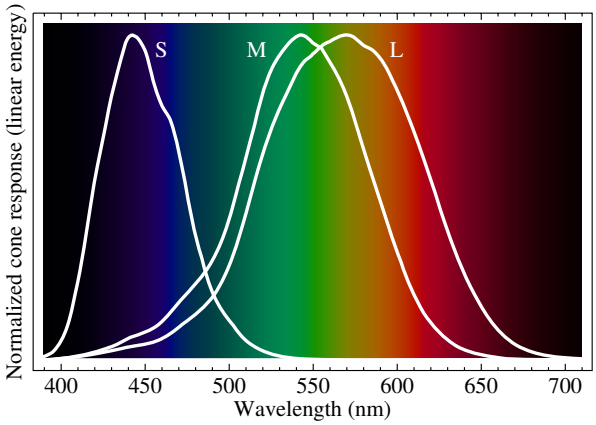


FIGURA 3.5: Respuesta normalizada de los conos tipo S, M y L del ojo humano [24]. Longitud de onda en nanómetros (nm).

Paleta de colores

Este formato genera 256 colores diferentes en pantalla. La paleta resultante se muestra en figura 3.6. Esta paleta no se puede modificar o redefinir ya que el color se asigna directamente sobre cada pixel y las combinaciones posibles responden a la cantidad de bits de cada componente.

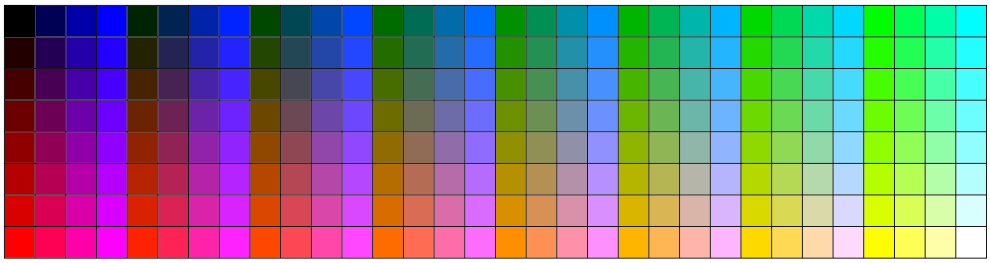


FIGURA 3.6: Paleta de colores del formato de pixel RGB 3:3:2.

Pixmaps

En la figura 3.7 puede apreciarse una foto convertida automáticamente a pixmap RGB 3:3:2 con y sin «ruido» agregado (Floyd-Steinberg dithering).

FIGURA 3.7: Ejemplo de conversion de fotos a RGB 3:3:2 (B) Foto original. (A) RGB 3:3:2 con dither. (C) RGB 3:3:2 sin dither. Foto gentileza de Kodak.

Los pixmaps utilizados en este tipo de sistemas suelen dibujarse a mano pixel a pixel para aprovechar al máximo la paleta de colores y resolución disponible. Esta técnica se llama «Pixel Art» y produce resultados como se ve en la figura 3.8

FIGURA 3.8: Ejemplo de pixmaps realizados con la técnica de Pixel Art. Imagen de 256x144 pixels en formato RGB 3:3:2. Captura de pantalla modificada para formato 16:9 del videojuego *Wonder Boy in Monster World* (1991) para consola SEGA Mega Drive.

Requisitos de hardware

Se requiere de dos tipos de memoria en cantidad suficiente:

1. Almacenamiento de pixmaps en memoria no volatil (Flash, EEPROM).
2. Memoria RAM de uso exclusivo del Framebuffer.

Como referencia, la imagen de la figura 3.8 ocupa $256 \times 144 \times 1\,\mathrm{B} = 36\,\mathrm{KiB}$.

Para generar las señales de video se requieren 8 salidas digitales de alta velocidad.

3.2.7. Modo de video

Se utiliza el mínimo soportado por el estándar HDTV: 1280x720 pixeles con frecuencia de actualización de 60 Hz y abreviado como 720p@60 Hz. En la figura 3.9 puede observarse una comparativa entre 720p y VGA 640x480.

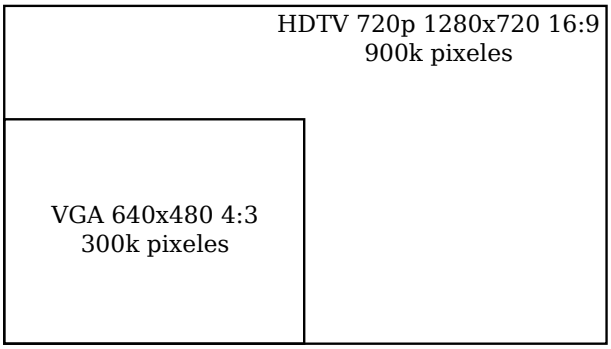


FIGURA 3.9: Comparativa de tamaño, cantidad de pixeles y relación de aspecto entre HDTV 720p y VGA.

A continuación se enumeran los fundamentos de esta elección.

Relación de aspecto

Se desea utilizar el 100 % de la pantalla visible evitando las bandas negras a los costados de una imagen 4:3 en una pantalla «ancha» de 16:9.

Compatibilidad

Asegurar compatibilidad con todos los televisores y monitores de tipo LCD o similar manufacturados desde el 2005 en adelante.

Los televisores LCD continuaron soportando modos de video legados o incluso arbitrarios dentro de su rango de frecuencias permitidas, pero la interfaz de video dominante HDMI (High-Definition Multimedia Interface) en versiones 1.0 y 1.1 solo soporta SDTV[7](480i, 576i) o HDTV (720p, 1080i, 1080p) [16]. Respetar esta limitación es importante a fin de asegurar buena conectividad.

FIGURA 3.10: Cable de interconexión HDMI Tipo A.

Escalado y calidad de imagen

Los televisores y monitores LCD tienen una resolución nativa, esto es, una cantidad fija de pixeles en horizontal y vertical de su pantalla visible y deben aplicar un algoritmo de escalado si la señal no coincide con su resolución nativa, lo que muchas veces distorsiona la imagen original.

A menor resolución de imagen en la señal de video, mayor es el efecto del algoritmo de escalado, lo que produce imágenes suavizadas, difuminadas o «aguadas»[8]como puede verse en la figura 3.11.

Generar una señal de video HDTV da como resultado un escalado mínimo o imperceptible, con menor degradación de imagen. Esto es particularmente importante si el contenido a visualizar es de baja resolución.

3.2.8. Framebuffer

Se revisó la disponibilidad de memoria RAM en el microcontrolador de la EDU-CIAA-NXP a fin de comprender cuales son las restricciones de diseño.

[7] *Standard Definition Television* Televisión de Definición Estandar. Hace referencia a los modos de video derivados de los antiguos estándares analógicos «NTSC» y «PAL»

[8] El efecto de aplicar un algoritmo de escalado basado en interpolación lineal es similar al de un filtro pasa bajos.

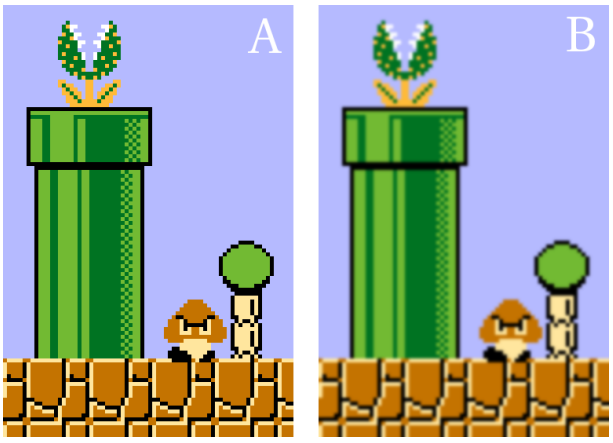


FIGURA 3.11: Simulación del efecto de escalado en televisores LCD. (A) Fragmento de 74x110 de una señal de SDTV de una consola «Family Game» visto en un monitor CRT. (B) Mismo contenido en una pantalla HDTV. Imagen del videojuego *Super Mario Bros.* (1985) para consola Nintendo Famicom.

El integrado LPC4337JBD144 dispone de 136 KiB de memoria SRAM [23, p.21]

En la figura 3.12 se observa que dicha memoria esta dividida en cuatro bloques de diferentes tamaños.

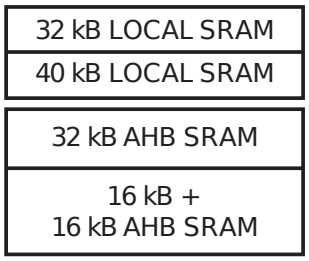


FIGURA 3.12: Memoria SRAM en diagrama de bloques del microcontrolador LPC4337 [23, p.25].

En la figura 3.13 se indica que, de los cuatro bloques disponibles, solo los denominados «AHB» están posicionados en direcciones de memoria contigua.

Part	Local SRAM	Local SRAM	M0 subsystem SRAM	AHB SRAM	AHB SRAM	AHB SRAM/ ETB SRAM
	0x1000 0000	0x1008 0000	0x1800 0000	0x2000 0000	0x2000 8000	0x2000 C000
LPC433x	32 kB	40 kB	-	32 kB	16 kB	16 kB

FIGURA 3.13: Bloques de memoria SRAM en microcontrolador LPC4337 [23, p.36].

Resumiendo, las posibilidades de memoria contigua total son:

- 32 KiB RamLoc32 en dirección 10 00 00 00 h.
- 40 KiB RamLoc40 en dirección 10 08 00 00 h.

- 64 KiB conjunto «RamAHB» en dirección 20 00 00 00 h.

El conjunto «RamAHB» de 64 KiB esta conformado por el bloque de 32 KiB RamAHB32 en 20 00 00 00 h, 16 KiB RamAHB16 en 20 00 80 00 h y 16 KiB RamAHB_ETB16[9] en 20 00 C0 00 h.

El grupo de bloques «RamAHB» seria el limite maximo de tamaño de un solo Framebuffer. Sin embargo en la Sección 3.2.4 se decidió utilizar Doublebuffer. De ahí que el segundo bloque mas grande, RamLoc40 de 40 KiB, es el limite máximo.

Fijado el limite máximo se procedió a la elección de la resolución lógica del Framebuffer, esto es el tamaño del área de pixeles disponibles en donde efectivamente se dibuja.

Por las restricciones de memoria de este trabajo, la resolución lógica necesariamente debe ser diferente al modo de video que utiliza el adaptador para generar la señal.

En la Tabla 3.1 se listan diferentes resoluciones múltiplo del modo de video y los tamaños de Framebuffer requeridos. Como se observa en esa tabla, con las restricciones de memoria y el requerimiento de Doublebuffer, la mejor (y casi única) alternativa es utilizar una resolución lógica de 256x144. Afortunadamente, ese numero se acerca mucho a la resolución de las consolas de videojuegos clásicas.

TABLA 3.1: Resoluciones múltiplo del modo de video y tamaños de Framebuffer.

Resolución	Relación con modo de video	Tamaño
1280x720	$(1/1) : (1/1)$	900 KiB
640x360	$(1/2) : (1/2)$	225 KiB
320x180	$(1/4) : (1/4)$	56,25 KiB
256x144	$(1/5) : (1/5)$	36 KiB

Como se decidió que la resolución lógica del Framebuffer fuese $(1/5) : (1/5)$ de la resolución del modo de video, se le asigna al adaptador de video la tarea de escalar la información presente en el Framebuffer.

Esta técnica es ideal para emular la estética de los videojuegos antiguos de baja resolución[10] en pantallas HDTV. Es el adaptador de video del sistema (y no la pantalla LCD, ver Sección 3.2.7) quien se ocupa de mapear perfectamente, sin filtros ni distorsiones, cada pixel del Framebuffer en exactamente 5x5 pixeles en pantalla, garantizando la fidelidad y consistencia de la imagen a través de diferentes marcas y modelos de pantalla.

Las dos áreas de memoria idénticas y correlativas para implementar Doublebuffer se ubicaron las siguientes áreas de memoria:

- 36 KiB de RamLoc40 en dirección 10 08 00 00 h para framebufferA.
- 32 KiB de RamAHB32 en dirección 20 00 00 00 h para framebufferB
- 4 KiB de RamAHB16 en dirección 20 00 80 00 h para framebufferB

[9] ETB (*Enhanced Trace Buffer*) es una función opcional para el bloque que este trabajo no utilizará

[10] La resolución lógica en consolas como «Family Game» es de 256x240 pixeles.

El total de la memoria SRAM utilizada para video es de 72 KiB. La memoria disponible para otros usos es de 64 KiB.

3.2.9. Interfaz de video

En esta subsección se revisarán los criterios usados para elegir una interfaz de video óptima según los objetivos, requerimientos y restricciones de este trabajo.

Actualmente existen varias interfaces de video con diferentes grados de soporte según el año de fabricación de la pantalla y si la pantalla es un televisor o monitor de PC. Todas las interfaces en el siguiente listado soportan el modo de video de 1280x720@60 Hz.

HDMI Es una interfaz digital y actualmente la mas soportada en televisores y monitores. Utiliza señales TMDS[11]de alta velocidad, imposibles de implementar con el microcontrolador. Un integrado como el TFP410 de Texas Instruments aportaría la codificación TMDS, pero su costo es prohibitivo para este trabajo.

FIGURA 3.14: Conector HDMI Tipo A.

VGA Es la interfaz de video para PC mas utilizada en los 90's y 2000' y existe en todos los televisores LCD fabricados hasta 2015, año en el cual se dejó de soportar. A diferencia de video compuesto, esta interfaz transmite los diferentes datos de video sin codificar mediante conductores independientes; a esto se le llama una señal de video «por componentes». En este sentido es similar a YPbPr o SCART RGB[12]. Los componentes de la señal de video son los colores rojo, verde y azul mas el sincronismo vertical y el sincronismo horizontal y se abrevia como «RGBHV». Implementarla con el microcontrolador solo requiere el uso de salidas digitales, buffers y resistencias. El conector puede verse en la figura 3.15.

FIGURA 3.15: Conector VGA.

YPbPr Se encuentra presente en todo televisor LCD e incluso en muchos de los últimos tevisores de tubo. Los componentes se dividen en Y+Sync (luminancia mas sincronismo), Pb (diferencia entre B e Y) y Pr (diferencia entre R e Y). Al ser analógica por componentes su implementación es igual de simple y barata que VGA. Pero convertir del espacio RGB al YPbPr *al vuelo* es una operación costosa para el adaptador de video. Para salvar esta restricción seria necesario que el framebuffer y los pixmaps no estén definidos en el espacio de color RGB, sino en YPbPr. La conexión se realiza a través

[11] *Transition-minimized differential signaling*. Par diferencial con codificación 8b/10b.

[12] Norma europea cuyo conector soporta video compuesto y también video por componentes RGB.

de tres conectores RCA como se ve en la figura 3.16. Ningún monitor de PC soporta YPbPr.

FIGURA 3.16: Conectores YPbPr.

DVI Tiene versiones en digital (DVI-D, TMDS compatible con HDMI), analógica (DVI-A, igual a VGA) y soporte de ambas (DVI-I). Los televisores solo soportan DVI-D mediante conector HDMI. Muchos monitores soportaban DVI-D via el conector original DVI, pero luego la industria se inclinó a favor de HDMI o de estándares digitales mas nuevos como DisplayPort.

FIGURA 3.17: Conector DVI-I.

Luego de evaluar las posibilidades técnicas y de costo de cada opción, se decidió adoptar la interfaz de video VGA.

Se tuvo en cuenta que Intel y AMD dejaron de soportar VGA en sus chipsets a partir del año 2015 y que se removió la interfaz en los nuevos diseños de televisores. Sin embargo, todavía existe en uso una gran cantidad de televisores y monitores LCD que soportan VGA de forma nativa. Ademas hay disponibilidad de adaptadores de interfaz VGA a HDMI como se ve en la figura 3.18 que son económicos y en general brindan excelente calidad de imagen y muy buena compatibilidad.

FIGURA 3.18: Conversor de interfaz de video VGA a HDMI.

3.2.10. Señal de video

En esta subsección se detalla la definición de la señal de video analógica para el modo elegido de 1280x720@60 Hz.

Introducción

La señal de video analógica por componentes está íntimamente ligada al principio de funcionamiento de los hoy obsoletos monitores CRT. Este tipo de dispositivo consiste en un tubo de vidrio al vacío. La cara interna de la pantalla visible esta recubierta de fósforo. Un haz de electrones se dispara desde el fondo de la pantalla y recorre cada linea de izquierda a derecha y de arriba hacia abajo de la imagen, impactando en un punto determinado del recubrimiento con una energía proporcional al brillo del pixel y produciendo que el fósforo emita luz en ese punto [19, p.55]. Este movimiento de escaneo para visualizar una imagen se denomina *Raster Scan* [3, p.12] y cada linea en este proceso es una *Scanline*.

El aviso de nueva linea se denomina «Horizontal Sync» o *HSYNC* y el aviso de nueva imagen, «Vertical Sync» o *VSYNC*. En estos periodos de sincronismo no se dibujan pixeles. La figura 3.19 ilustra este proceso.

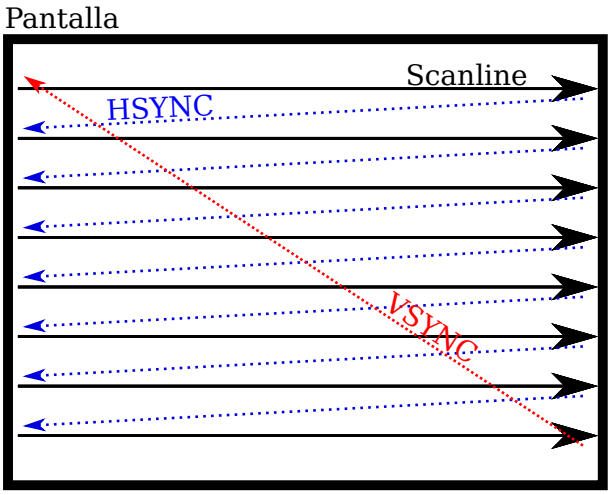


FIGURA 3.19: Proceso de Raster Scan en una pantalla CRT.

Un momento antes de la señal de sincronismo se deben apagar los pixeles visibles y después de la misma, dar tiempo a que la electrónica del CRT reposicione el haz de electrones y este listo para dibujar nuevamente. Por esto, antes de la señal de sincronismo existe un tiempo de espera llamado *Front Porch* e inmediatamente después, otro llamado *Back Porch*.

Al periodo de tiempo conformado por *Front Porch*, *Sync* y *Back Porch* se le denomina *Blanking Interval*. Existe un «Horizontal Blanking Interval» y un «Vertical Blanking Interval».

Se conoce como «activo» al periodo de tiempo en donde se dibujan pixeles. Esencialmente es en todo momento excepto en los periodos de *Blanking Interval*.

En los CRT la señal de video es la que efectivamente comanda el haz de electrones e imprime cada pixel en la pantalla de manera casi instantánea. En cambio las pantallas LCD disponen de un buffer de imagen y solo actualizan su pantalla luego de haber recibido la imagen completa.

Temporizado

Para generar la señal de video es necesario definir el temporizado de todos los elementos que la componen.

En un primer momento se pensó en copiar el temporizado de una señal de video HDTV 720p conocida. Por esta razón el requerimiento 4.4.3 del *Plan de proyecto* habla de que «El sincronismo vertical y horizontal serán positivos» [6, p.10]. Pero en la etapa de experimentación esto no dio resultados óptimos y se decidió generar el temporizado utilizando un estándar VESA[13]: *Coordinated Video Timings* o CVT.

El estándar CVT es una formula que toma en cuenta tiempos mínimos de Blanking en un CRT y genera temporizados con resoluciones arbitrarias pero sincronismos compatibles a través de dispositivos.

En Linux existe una aplicación llamada «cvt» que implementa esta formula y se utiliza para generar lineas de temporizado en el archivo de configuración de xorg-server[14]. En el listado 3.1 puede verse el manual de la aplicación.

LISTADO 3.1: Pagina de manual (manpage) del comando CVT

```
CVT(1)                      General Commands Manual                     CVT(1)

NAME
       cvt - calculate VESA CVT mode lines

SYNOPSIS
       cvt [-v|-verbose] [-r|-reduced] h-resolution v-resolution [refresh]

DESCRIPTION
       Cvt  is  a utility for calculating VESA Coordinated Video Timing modes.
       Given the desired  horizontal  and  vertical  resolutions,  a  modeline
       adhering  to the CVT standard is printed. This modeline can be included
       in Xorg xorg.conf(5)

OPTIONS
       refresh Provide a vertical  refresh  rate  in  Hz.   The  CVT  standard
               prefers  either  50.0,  60.0,  75.0  or 85.0Hz.  The default is
               60.0Hz.

       -v|-verbose
              Warn verbosely when a given mode does not completely correspond
              with CVT standards.

       -r|-reduced
              Create  a  mode  with reduced blanking.  This allows for higher
              frequency signals, with a lower  or  equal  dotclock.  Not  for
              Cathode Ray Tube based displays though.

SEE ALSO
       xorg.conf(5), gtf(1)

AUTHOR
       Luc Verhaegen.

       This  program is based on the Coordinated Video Timing sample implemen-
       tation written by Graham Loveridge. This file is publicly available  at
       <http://www.vesa.org/Public/CVT/CVTd6r1.xls>. CVT is a VESA trademark.
```

[13]VESA es *Video Electronics Standards Association*, una asociación de compañías que publican estándares para la industria de los dispositivos de video.

[14]El sistema cliente-servidor para interfaz gráfica de usuario usado en los diferentes «escritorios» de Linux.

```
38
39 X Version 11                xorg-server 1.19.2                CVT(1)
```

Se llama a la aplicación pasando como parámetro la resolución activa y frecuencia de refresco deseados para obtener un temporizado compatible con CVT como se muestra en el listado 3.2.

LISTADO 3.2: Uso del comando CVT para generar temporizado.

```
1 usuario@linux:$ cvt --verbose 1280 720 60
2 # 1280x720 59.86 Hz (CVT 0.92M9) hsync: 44.77 kHz; pclk: 74.50 MHz
3 Modeline "1280x720_60.00"   74.50  1280 1344 1472 1664  720 723 728 748 -hsync +
4 vsync
```

La aplicación devuelve el temporizado en formato *Modeline* para incluir directamente en el archivo de configuración de xorg-server. En la tabla 3.2 se analizan los parámetros del *Modeline* según su significado.

TABLA 3.2: Significado de los parámetros del *Modeline* generado por CVT.

Valor	**Nombre (xorg)**	**Significado**
Modeline		Inicia definición de modo de video en xorg.conf
"1280x720_60.00"	mode name	Nombre de la definición
75.50	dot clock	Pixel clock, frecuencia de pixel, 75,50 MHz
1280	hdisp	Pixeles activos en horizontal
1344	hsyncstart	En que pixel comienza señal de sincro. horizontal
1472	hsyncend	En que pixel termina señal de sincro. horizontal
1664	htotal	Total de pixeles en horizontal (activos + blanking)
720	vdisp	Pixeles activos en vertical
723	vsyncstart	En que linea comienza señal de sincro. vertical
728	vsyncend	En que linea termina señal de sincro. vertical
748	vtotal	Total de lineas verticales (activas + blanking)
-hsync		Señal de sincro. horizontal con polaridad negativa
+vsync		Señal de sincro. vertical con polaridad positiva

En la figura 3.20 se observa un diagrama de la señal de video según los parámetros devueltos por la aplicación CVT.

El modo de video resultante tiene un total de 1664x748 pixeles contando los intervalos de Blanking. Cada cuadro de video se actualiza a una frecuencia de $59{,}86\,\mathrm{Hz}$ y equivale a la frecuencia de sincronismo vertical (*VSYNC*). La frecuencia de sincronismo horizontal ($\overline{HSYNC}$) es de $44{,}77\,\mathrm{Hz}$ y la de pixel (*PCLK*) es $74{,}50\,\mathrm{MHz}$.

Las ecuaciones en 3.3 (*PCLK*), 3.4 ($\overline{HSYNC}$) y 3.5 (*VSYNC*) corroboran estos datos e ilustran sobre la relación existente entre los parámetros.

$$PCLK = 1664 \cdot 748 \cdot 59{,}86\,\mathrm{Hz} = 74\,506\,065{,}92\,\mathrm{Hz} = 74{,}50\,\mathrm{MHz} \tag{3.3}$$

$$\overline{HSYNC} = \frac{74\,506\,065{,}92\,\mathrm{Hz}}{1664} = 44\,775{,}28\,\mathrm{Hz} = 44{,}77\,\mathrm{kHz} \tag{3.4}$$

$$VSYNC = \frac{74\,506\,065{,}92\,\mathrm{Hz}}{1664 \cdot 748} = 59{,}86\,\mathrm{Hz} \tag{3.5}$$

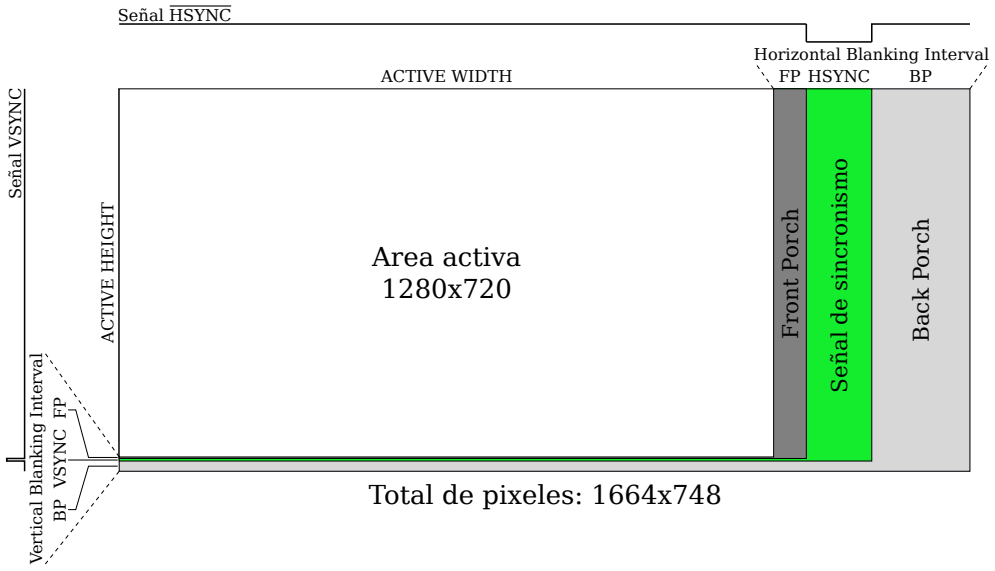


FIGURA 3.20: Diagrama de la señal de video calculada por la aplicación CTV.

En la ecuación 3.6 (*TPIXEL*) se calcula el periodo de un solo pixel.

$$TPIXEL = \frac{1}{74\,506\,065{,}92\,\text{Hz}} = 1{,}342\,172\,597\,16 \cdot 10^{-8}\,\text{s} = 13{,}42\,\text{ns} \tag{3.6}$$

Utilizando el valor de (*TPIXEL*), en la tabla E.2 se calcula la duración del Front Porch, sincronismos, Back Porch y demás tiempos que serán útiles en la implementación del driver del adaptador de video.

TABLA 3.3: Intervalos de tiempo en la señal de video. Números truncados al quinto decimal.

Parámetro	**Intervalo**	**Cantidad**	**Tiempo**
Horizontal Total (Line)	1664_{htotal}	1664 pixeles	22,333 75 µs
Horizontal Active Pixels	1280_{hdisp}	1280 pixeles	17,179 80 µs
Horizontal Blanking Interval	$1664_{\text{htotal}} - 1280_{\text{hdisp}}$	384 pixeles	5,153 94 µs
Horizontal Front Porch	$1344_{\text{hsyncstart}} - 1280_{\text{hdisp}}$	64 pixeles	0,858 99 µs
Horizontal Sync Width	$1472_{\text{hsyncend}} - 1344_{\text{hsyncstart}}$	128 pixeles	1,717 98 µs
Horizontal Back Porch	$1664_{\text{htotal}} - 1472_{\text{hsyncend}}$	192 pixeles	2,576 97 µs
Vertical Active Lines	720_{vdisp}	720 lineas	16,080 30 ms
Vertical Blanking Interval	$748_{\text{vtotal}} - 720_{\text{vdisp}}$	28 lineas	0,625 34 ms
Vertical Front Porch	$723_{\text{vsyncstart}} - 720_{\text{vdisp}}$	3 lineas	0,067 00 ms
Vertical Sync Width	$728_{\text{vsyncend}} - 723_{\text{vsyncstart}}$	5 lineas	0,111 66 ms
Vertical Back Porch	$748_{\text{vtotal}} - 728_{\text{vsyncend}}$	20 lineas	0,446 67 ms

3.2.11. Interfaz eléctrica

La interfaz eléctrica es parte constitutiva de la interfaz de video elegida en la subsección 3.2.9. En esta subsección se revisará en detalle las señales eléctricas necesarias para implementar una interfaz de video VGA.

Conector

Para la interfaz de video VGA del lado del dispositivo, se utiliza un conector de tipo D-Sub de alta densidad, hembra y 15 posiciones. En la figura 3.21 se observa el tipo de conector utilizado en este trabajo.

FIGURA 3.21: Conector de video utilizado en la RETRO-CIAA.

Disposición de pines

En la figura 3.22 se numeran las posiciones del puerto VGA (conector D-sub 15 de alta densidad hembra) visto de frente y se marcan algunas señales como referencia.

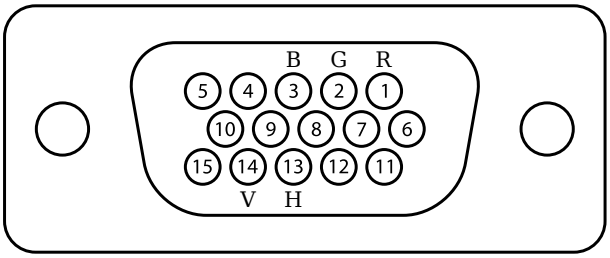


FIGURA 3.22: Numeración de posiciones del puerto VGA. Conector D-sub 15 hembra de alta densidad.

La disposición completa de señales del estándar VGA se encuentra en la tabla 3.4. Es importante aclarar que esta disposición responde a la incorporación de DDC2/E-DDC a fines de los '90. El estándar original no tiene relevancia para esta aplicación ya que toda pantalla contemporánea adhiere al ultimo estándar.

A continuación se describe en detalle de cada señal eléctrica que compone la interfaz.

Señales de color

Las posiciones 1, 2 y 3 transmiten las intensidades analógicas de los componentes de color rojo, verde y azul del pixel que se este transmitiendo en un momento dado.

Esta transmisión se realiza a través de tres lineas de cables coaxiales (uno para cada componente) de $75\,\Omega$ de impedancia. Esto es necesario ya que las señales cambian a una frecuencia del orden de decenas de mega-hertz.

Cada componente rojo, verde y azul tiene una posición de retorno particular en los pines 6, 7 y 8 respectivamente.

TABLA 3.4: Disposición de pines del estándar VGA en puerto D-sub 15 hembra de alta densidad.

Posición	Señal	Descripción
1	Red	Componente de color rojo
2	Green	Componente de color verde
3	Blue	Componente de color azul
4	Reserved	No utilizado
5	$\overline{\text{HSYNC}}$ Ground	Retorno de la señal $\overline{\text{HSYNC}}$
6	Red Ground	Retorno de la señal Red
7	Green Ground	Retorno de la señal Green
8	Blue Ground	Retorno de la señal Azul
9	PWR	Brinda $5\,\text{V}$ al EDID EEPROM del monitor
10	VYSNC, DDC Ground	Retorno de VSYNC y PWR
11	Reserved	No utilizado
12	SDA	Datos I^2C para DDC2
13	HSYNC	Señal de sincronismo horizontal
14	VSYNC	Señal de sincronismo vertical
15	SCL	Reloj I^2C para DDC2

Idealmente la tensión de la señal analógica debe tener una amplitud de $0\,\text{V}$ a $0{,}7\,\text{V}$ en cada extremo de la linea de transmisión.

Señales de sincronismo

Las posiciones 13 y 14 transmiten las señales de sincronismo horizontal y vertical respectivamente. Son señales TTL de $5\,\text{V}$ y de baja a media frecuencia. En este trabajo las frecuencias aproximadas son $60\,\text{Hz}$ para VSYNC y $45\,\text{kHz}$ para $\overline{\text{HSYNC}}$.

Disponen de un retorno especifico en los pines 5 para $\overline{\text{HSYNC}}$, y 14 para VSYNC.

Señales de comunicación

DDC o *Display Data Channel* es un protocolo estandarizado por VESA para permitir que la pantalla comunique modos de video soportados al adaptador de video a fin de que este ultimo ajuste automáticamente los parámetros de la señal [19, p.207].

El objetivo del estándar fue generar una experiencia «Plug and Play» sin tener que configurar modos compatibles a mano o instalar drivers específicos.

La versión 1 de DDC de 1994 no fue adoptada en forma masiva.

La versión 2 de DDC de 1996 utiliza el protocolo I^2C para que el adaptador, que siempre actuá como master I^2C, se comunique con una EEPROM en la pantalla y obtenga $128\,\text{B}$ de datos con información de modos soportados.

E-DDC es una extensión de DDC2. Soporta lectura de EEPROMs de hasta $32\,\text{KiB}$ mediante acceso segmentado [25].

Las posiciones 12 y 15 transmiten las señales SDA y SCL del protocolo I^2C. La velocidad máxima de comunicación es de $100\,\text{kHz}$ [19, p.208].

La posición 9 (PWR) esta destinada a alimentar la EEPROM de la pantalla. La posición 10 se dispone para el retorno de estas señales.

Asignación de pines del microcontrolador

Los pines de E/S del microcontrolador se dividen en puertos y se identifican por un numero: GPIO0, GPIO1, GPIO2, etc. [26]. La cantidad de pines de un GPIO esta determinado por el encapsulado del microprocesador [23, p.458], siendo el máximo lógico 32 pines por puerto correspondiente a los 32 bit de palabra de la arquitectura.

Se necesitan 8 pines de GPIO para generar las señales de los componentes de color RGB 3:3:2. Es necesario que estos 8 pines cambien al mismo tiempo de forma sincrónica ya que todos son componentes de color de un mismo pixel.

Existen dos formas de escribir en un pin:

Puerto completo Escribiendo una palabra (32 bit) en la dirección de memoria que contiene el estado de un puerto entero. Se accede simultáneamente a todos pines del mismo puerto.

Pin individual Escribiendo un byte en un área de memoria especial para un pin especifico. Toda escritura en esas áreas especiales se traducen de byte a bit y se asignan al pin correspondiente. En la arquitectura del microcontrolador esto se denomina «Bit banding». Este modo permite cambiar un pin particular de manera atómica sin realizar una lectura, mascara, modificación y escritura del puerto entero. Solo se puede modificar un pin a la vez.

Escribir en diferentes puertos completos o en diferentes pines individuales produce un retardo en cada acceso que no debe despreciarse. En consecuencia, se necesita que los 8 pines pertenezcan al mismo puerto. Ademas sería optimo que los pines sean correlativos de modo que la asignación de bits a pines al momento de escribir en el puerto genere un mapeo directo entre un pixel en memoria y sus señales eléctricas.

Lamentablemente los puertos de expansión de la EDU-CIAA-NXP no poseen pines con todas estas características. En la tabla 3.5 puede verse la mejor disposición de pines que se encontró considerando las restricciones.

TABLA 3.5: Asignación de señales de componentes de color a puerto y pin en LPC4337.

Señal	Puerto	Pin
R2	GPIO2	8
R1	GPIO2	6
R0	GPIO2	5
G2	GPIO2	4
G1	GPIO2	3
G0	GPIO2	2
B1	GPIO2	1
B0	GPIO2	0

La ubicación no ideal de la señal R2 en el bit 8 del GPIO2 implica que no se pueda realizar una asignación directa del pixel en memoria a la dirección del puerto como hubiese sido optimo, sino que se requiere un paso extra para acomodar el bit como puede verse en la figura 3.23

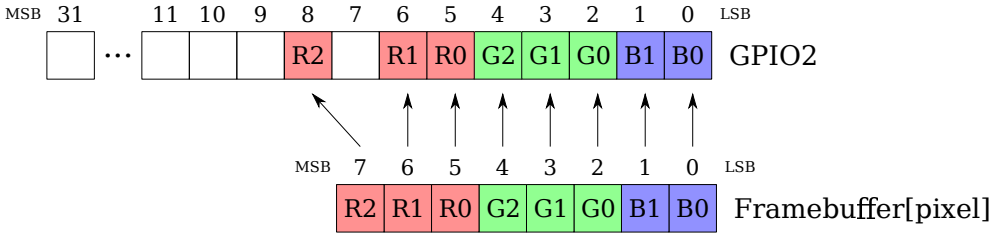


FIGURA 3.23: Disposición de bits en el puerto GPIO2 de señales eléctricas RGB y en un pixel en el framebuffer.

De todos modos, una vez realizado el acomodamiento del bit en un registro auxiliar, la asignación de los bits y cambio de los pines de GPIO se ejecutan todos al mismo tiempo.

Al asignar un valor de pixel no se deben tocar los bits del puerto GPIO2 que no pertenecen a las señales de color. Esto implica no tocar GPIO2[7] y tampoco GPIO2[9] a GPIO2[31]. Para ese fin el microcontrolador dispone de un mecanismo para asignar a un puerto una mascara de los bits «modificables» (dirección MASK2) y un área en memoria donde escribir utilizando esa mascara (dirección MPORT2) [23, p.472].

En la tabla 3.6 se listan las direcciones de acceso de MASK2, MPORT2 y el valor de la mascara calculada para modificar solamente los pines de las señales de color en GPIO2.

TABLA 3.6: Datos útiles para asignar valores a señales de color.

Parametro	Valor	Comentario
MASK2	40 0F 60 88 h	Dirección de MASK2 en bloque high-speed GPIO.
MPORT2	40 0F 61 88 h	Dirección de MPORT2 en bloque high-speed GPIO.
Mascara	FF FF FE 80 h	Asignar a MASK2 antes de asignar pixel a MPORT2.

MASK2 y MPORT2 pertenecen al bloque de memoria reservado para el «high-speed GPIO» en la dirección base 40 0F 40 00 h [23, p.463].

Generación de señales

Todas las señales de video se generan desde el núcleo M0 del microcontrolador de la EDU-CIAA-NXP, se emiten mediante pines de GPIO, se protegen mediante buffers y sus características eléctricas se adaptan con componentes pasivos para finalmente salir por el conector de video.

En la tabla 3.7 se observan las señales de video y a que pines del puerto de expansión de la EDU-CIAA-NXP corresponden.

A cada señal binaria de componentes de color y de nivel de 3,3 V se le aplica una resistencia serie de modo que, al unirse todas eléctricamente en un único pin de salida por color, su contribución al nivel de tensión total sea proporcional a su peso en bits. En la tabla 3.8 se listan los valores de resistencia serie correspondiente a cada componente de color. Los valores tienen una tolerancia máxima de 1 %.

TABLA 3.7: Señales de video, pines de expansión en la EDU-CIAA-NXP y pin/puerto y periférico en el microcontrolador LPC4337.

Señal	EDU-CIAA-NXP	LPC4337JBD144	Descripción
R2	GPIO8	P6_12, GPIO2[8]	Componente rojo, bit 2 (MSB)
R1	LCD3	P4_6, GPIO2[6]	Componente rojo, bit 1
R0	LCD2	P4_5, GPIO2[5]	Componente rojo, bit 0 (LSB)
G2	LCD1	P4_4, GPIO2[4]	Componente verde, bit 2 (MSB)
G1	TEC_F3	P4_3, GPIO2[3]	Componente verde, bit 1
G0	TEC_F2	P4_2, GPIO2[2]	Componente verde, bit 0 (LSB)
B1	TEC_F1	P4_1, GPIO2[1]	Componente azul, bit 1 (MSB)
B0	TEC_F0	P4_0, GPIO2[0]	Componente azul, bit 0 (LSB)
$\overline{\text{HSYNC}}$	LCD_RS	P4_8, GPIO5[12]	Sincronismo horizontal
VSYNC	LCD4	P4_10, GPIO5[14]	Sincronismo vertical
EDDC_SCL	I2C_SCL	I2C0_SCL	Comunicación, reloj
EDDC_SDA	I2C_SDA	I2C0_SDA	Comunicación, datos
EDDC_EN	TEC_C2	P7_5, GPIO3[13]	Habilita buffer E-DDC

TABLA 3.8: Valor de resistencias serie en señales de componente de color. Tolerancia $1\,\%$

Señal	Valor R	Comentario
R0	$2\,\text{k}\Omega$	LSB
R1	$1\,\text{k}\Omega$	
R2	$499\,\text{k}\Omega$	MSB
G0	$2\,\text{k}\Omega$	LSB
G1	$1\,\text{k}\Omega$	
G2	$499\,\text{k}\Omega$	MSB
B0	$1\,\text{k}\Omega$	LSB
B1	$499\,\text{k}\Omega$	MSB

Las resistencias fueron calculadas para llegar a la tensión máxima de $0{,}7\,\text{V}$ con todos los bits en 1 tomando en cuenta el efecto de la terminación paralela a masa de $75\,\Omega$ del lado de la pantalla.

La explotación de esa terminación paralela no trata a los cables coaxiales de cada componente como una linea de transmisión, pero es un truco de muy bajo costo utilizado en las salidas VGA de los kits de FPGA Xilinx modelos Nexys 2, Nexys 3 y Nexys 4 de Digilent Inc. Se decidió seguir este diseño luego de corroborar funcionamiento y calidad de imagen en una placa Nexys 4 con resolución de 1280x1024 y un cable de interconexión de $1{,}8\,\text{m}$ de largo.

A las señales digitales de sincronismo se les agrega una resistencia en serie de $100\,\Omega$ para reducir descargas o picos de tensión. A pesar de que eléctricamente VGA utiliza niveles TTL de 5V, el nivel de $3{,}3\,\text{V}$ utilizado en este trabajo es aceptable ya que TTL define un nivel alto a partir de los $2\,\text{V}$.

En el caso de las señales de comunicación DDC, las señales SCL y SDA de la EDU-CIAA-NXP se protegen y se cambian de nivel con un buffer I^2C, pasando de $3{,}3\,\text{V}$ a los $5\,\text{V}$ del estándar DDC. En el caso de I^2C si es importante respetar el nivel de $5\,\text{V}$ ya que a menor tensión e igual frecuencia, menor es la capacitancia y el largo de cable soportado. El integrado elegido para esta tarea es un TCA9617B de Texas Instruments en encapsulado VSSOP8. En la figura 3.24 puede verse un diagrama de conexión simplificado.

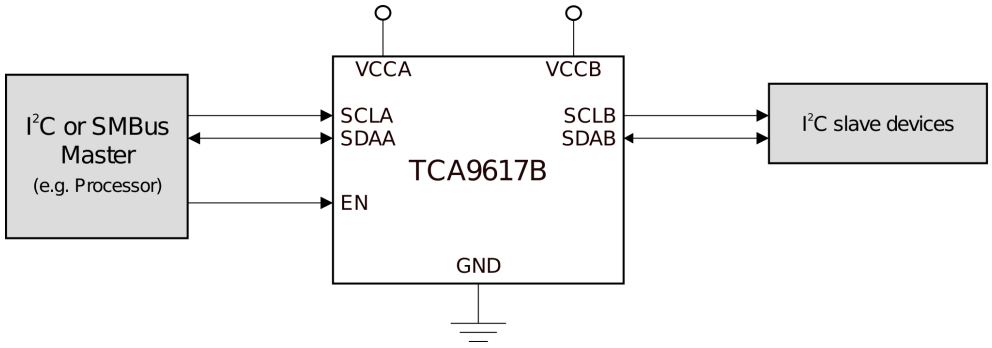


FIGURA 3.24: Diagrama de conexión simplificado del buffer I^2C TCA9617B.

En esta etapa de diseño no queda claro si es integrado TCA9617B es suficiente para soportar la capacitancia del cable VGA. De todos modos, el soporte de DDC es meramente experimental y no compromete ningún requerimiento del trabajo.

Cable de interconexión

El cable utilizado para conectar la pantalla al adaptador de video tiene dos conectores DB15 de alta densidad macho en cada extremo. Generalmente viene ensamblado de fabrica como puede verse en la figura 3.25. Se recomienda que el cable no posea una longitud mayor a $1{,}8\,\mathrm{m}$.

FIGURA 3.25: Cable de interconexión VGA.

3.2.12. Driver

El adaptador de video tiene un comportamiento definido por el driver de video implementado en firmware y ejecutado por el núcleo Cortex-M0.

El driver de este trabajo se encarga de las siguientes tareas:

- Generar la señal de video elegida.
- Obtener el contenido visible de la señal de video mediante el escalado en tiempo real de los pixeles en el «frontbuffer».
- Realizar un efecto retro de «scanlines CRT», también en tiempo real.

- Obtener el estado de los joysticks mientras se esta en el Veritical Blanking Interval.
- Avisar al núcleo Cortex-M4 cada vez que se ingresa al periodo de Vertical Blanking Interval.

En la figura 3.26 se observa un diagrama de flujo posible para el driver de video. El objetivo no es generar una implementación a partir de este gráfico, sino simplemente ilustrar la dirección de escaneo del frontbuffer, el funcionamiento del escalado y los intervalos de blanking.

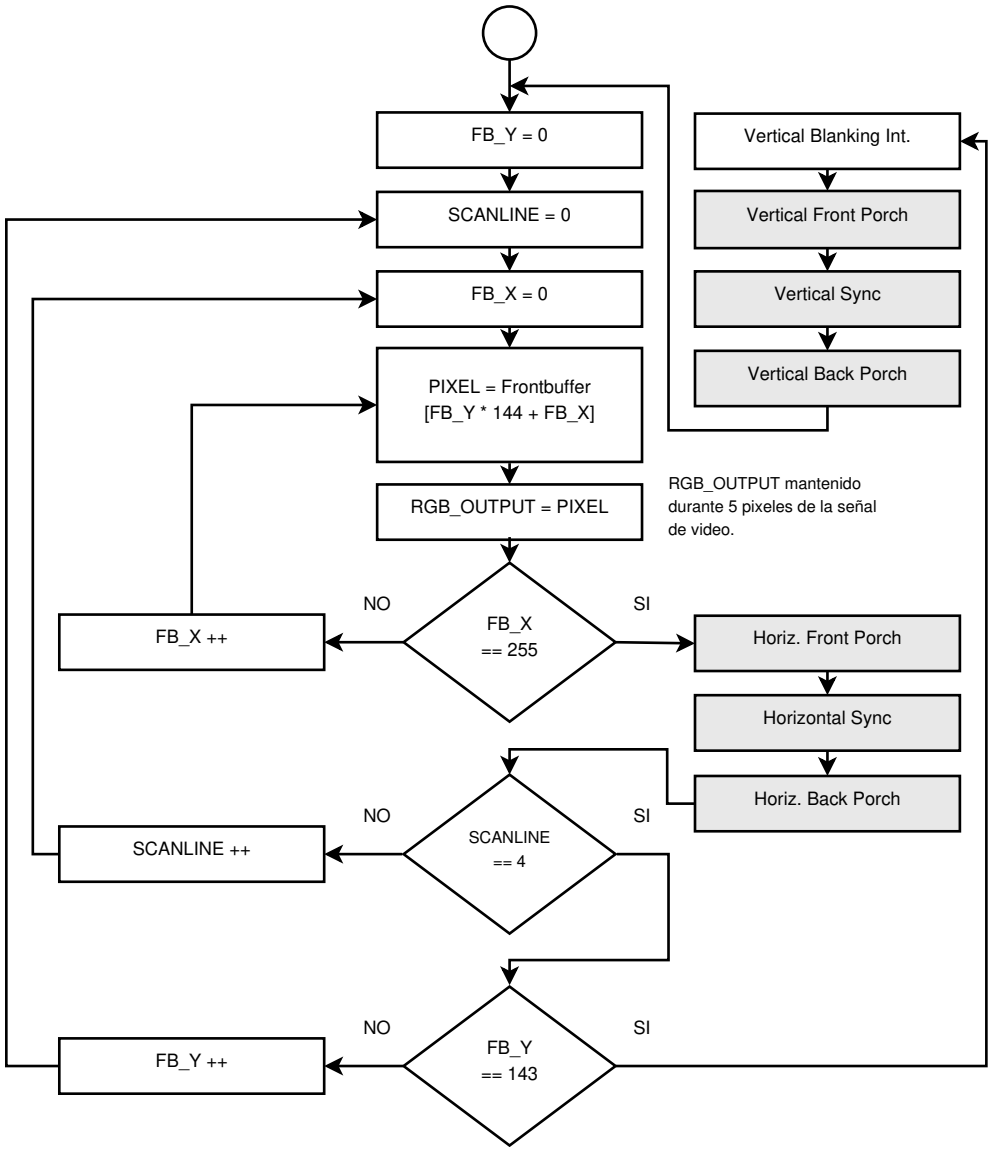


FIGURA 3.26: Diagrama de flujo de escalado y blankings en el driver de video.

Generación de la señal de video

Consisten en manejar los pines de salida del microcontrolador respetando las señales y los temporizados definidos en la señal de video elegida.

Escalado de contenido

El driver se ocupa de escalar y visualizar el contenido del «frontbuffer» mientras la aplicación dibuja en el «backbuffer». El proceso de escalado se ilustra en la figura 3.27.

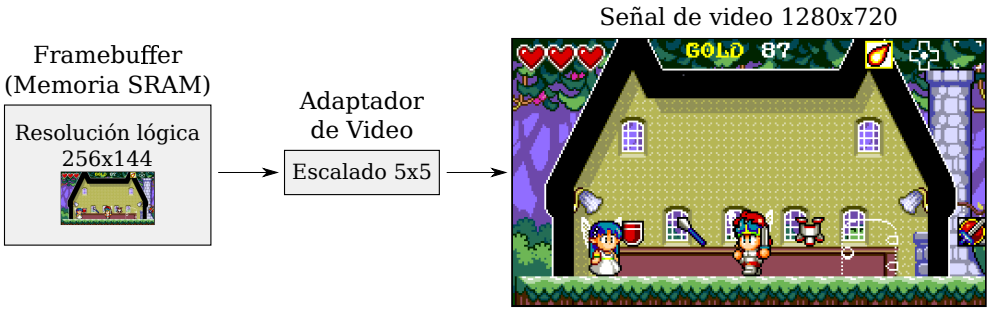


FIGURA 3.27: Proceso de escalado de Framebuffer a señal de video. Captura de pantalla modificada para formato 16:9 del videojuego *Wonder Boy in Monster World* (1991) para consola SEGA Mega Drive.

Un pixel lógico equivale a 5x5 pixeles iguales en la señal de video. Se lo divide verticalmente en 5 partes de este modo: 5x1, 5x2, 5x3, 5x4 y 5x5. Cada una de esas partes se dibuja en una linea de 720p o scanline y demora el tiempo especificado en la ecuación 3.7.

$$\begin{aligned} TLP5 &= TPIXEL \cdot 5 \\ &= 1{,}342\,172\,597\,16 \cdot 10^{-8}\,\mathrm{s} \cdot 5 \\ &= 6{,}710\,862\,985\,8 \cdot 10^{-8}\,\mathrm{s} = 67{,}108\,629\,858\,\mathrm{ns} \end{aligned} \tag{3.7}$$

Para realizar el escalado el adaptador emite un mismo color sobre la señal de video durante 5 pixeles (periodo *TLP5*) y el mismo scanline 5 veces seguidas. De este modo se dibuja en pantalla cada uno de los pixeles lógicos en 5x5 pixeles de la señal de video.

Efecto retro de «scanlines CRT»

En un dispositivo de escaneo raster los modos de video progresivos llenan todas las scanlines de manera progresiva, sin ningún salto entre lineas.

Los monitores CRT están preparados para utilizar modos de video entrelazados. Esto significa que las lineas o scanlines no se dibujan de modo progresivo sino que se dividen entre lineas impares (campo impar) y las pares (campo par).

En un modo de video de 480i@60 Hz se transmiten 60 campos por segundo de 240 lineas cada uno, 30 para el campo impar intercalados con 30 para el campo par. Las posiciones de linea impares se llenan con un campo de 240 lineas y luego las posiciones pares se llenan con otro campo de 240 lineas y así sucesivamente. La visualización de los pixeles se ejecuta de manera inmediata ni bien llega la información y la reconstrucción de la imagen es asistida por el «Fenómeno phi» en la percepción de quien mira.

El método de entrelazado funciona bien en televisión, en donde se mantiene la percepción de 480 lineas de resolución mientras permite ahorrar un 50 % de ancho de banda de transmisión de la señal. Pero no da buenos resultados como interfaz de imágenes generadas por computadora.

Las consolas de baja resolución evitan el entrelazado modificando la señal de video para indicarle al televisor que dibuje las 60 imágenes por segundo de 240 lineas en el mismo campo sin intercalado. Esto produce que el otro campo quede en negro, pero genera una resolución de 240p@60 Hz.

El efecto en la imagen resultante, tan característica de los videojuegos de baja resolución, se denomina comúnmente como «scanlines», haciendo referencia al espaciado de un scanline entre cada linea de contenido visible. Este efecto puede apreciarse en la figura 3.28. Notese que el brillo del fósforo del CRT puede iluminar parcialmente el campo que no se dibuja.

FIGURA 3.28: Efecto de «scanlines» de contenido 240p visualizado en un monitor CRT. Videojuego *Sonic the Hedgehog 2* (1992) para consola SEGA Mega Drive.

Desde el driver se implementa un efecto parecido dando la opción de indicar, mediante un numero entre 1 y 5 inclusive, cuantas de las 5 lineas que repiten una linea del frontbuffer deben efectivamente repetir esos datos o dibujarse en negro. Por ejemplo si se indica el numero 3, se repiten los pixeles del frontbuffer para las lineas 5x1, 5x2, 5x3 y las correspondientes a 5x4 y 5x5 se dibujan en negro.

Aviso de Vertical Blanking Interval

Este aviso se genera mediante una interrupción que se conoce como Vertical Blanking Interrupt. En este caso el núcleo Cortex-M0 le genera una interrupción al núcleo Cortex-M4 mediante la instrucción SEV.

Actualizar estado de joysticks

Para actualizar los joysticks es necesario generar un clock de baja frecuencia. El Cortex-M0 puede ocuparse de ello en los tiempos muertos de Vertical Blanking Interval, liberando al Cortex-M4 de realizar esta tarea.

3.3. Audio

El subsistema de audio se implementó utilizando un DAC de audio estéreo por protocolo I^2S.

3.3.1. Restricciones

Se encontró la siguiente restricción al encarar el diseño del subsistema de audio.

- La EDU-CIAA-NXP no se diseño para utilizar I^{2}S. No hay ninguna interfaz I2S con sus cuatro señales disponibles en los puertos de expansión.

3.3.2. Conceptos básicos

A continuación se presentan conceptos básicos de audio digital:

PCM «Pulse Code Modulation» es una forma de representar una señal de audio analógica en formato digital. Es el resultado de obtener muestras de la señal analógica a intervalos regulares (lo que define la *frecuencia* de cada muestra) y la amplitud de cada una es cuantificada, esto es, convertida a un valor entero finito en bits. Esto ultimo define los *bits por muestra* y junto a la frecuencia de muestreo determinan el formato de audio digital, por ejemplo audio digital de 16 bit y 44,1 kHz.

Estereo Se define un segundo canal de audio PCM. Un canal será el sonido proveniente de la izquierda y el otro canal el de la derecha. Mediante un parlante o altavoz dedicado a cada canal y dispuestos a izquierda y derecha del oyente, se genera sonido posicional. En el caso de PCM, es común que las muestras de ambos canales se guarden y transmitan intercaladas.

I^{2}S «Inter-IC Sound» es una interfaz eléctrica desarrollada por Philips Semiconductor (ahora NXP) para intercomunicar dispositivos de audio (circuitos integrados) de forma digital mediante datos PCM.

3.3.3. Elección del integrado

En los puertos de expansión de la EDU-CIAA-NXP se encuentra disponible la interfaz I2S1 sin su señal de *Master Clock*.

Existen integrados I^{2}S que derivan el *Master Clock* desde el clock de la señal de audio digital. La búsqueda necesariamente se centro en un integrado que disponga de esa capacidad.

El integrado PCM5100A de Texas Instruments posee esta característica necesaria y es de amplia disponibilidad. El encapsulado TSSOP20 que puede verse en la figura 3.29 es sencillo de ensamblar a mano en los prototipos.

FIGURA 3.29: DAC de audio PCM5100A de Texas Instruments.

En la figura 3.30 se observa el diagrama de bloques del PCM5100A. Las entradas I^{2}S de este integrado se denominan DIN, LRCK, BCK y SCK.

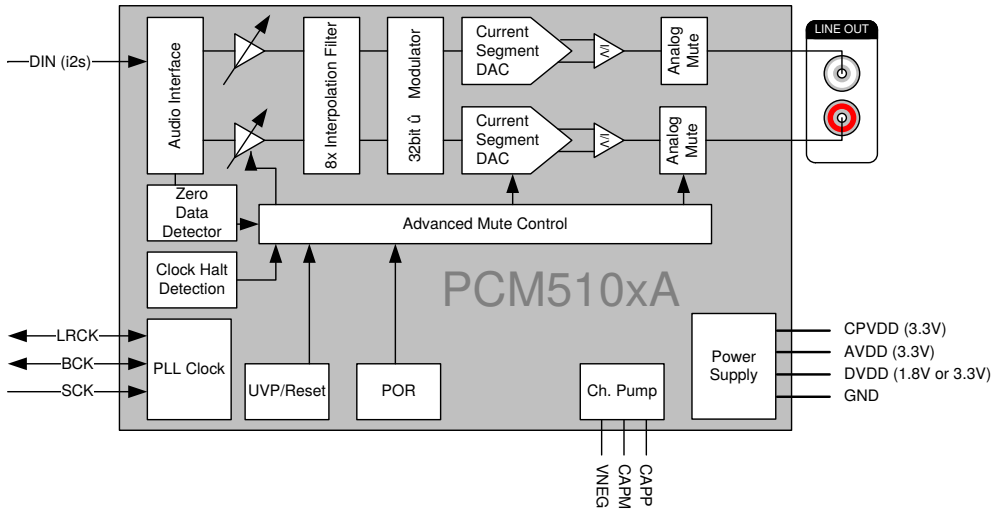


FIGURA 3.30: Diagrama de bloques del integrado PCM5100A.

3.3.4. Señales de control

El integrado se pone en estado de silencio o «mute» mediante la señal $\overline{\text{AUDIO\_MUTE}}$. En la tabla 3.10 se lista dicha conexión.

TABLA 3.9: Señal de control entre PCM5100A y EDU-CIAA-NXP.

Señal	PCM5100A	EDU-CIAA-NXP	Descripción
$\overline{\text{AUDIO\_MUTE}}$	XSMT	ETH_MDC	Salida en silencio (Activo en nivel bajo)

3.3.5. Señales I²S

En la tabla 3.10 se listan las señales I²S del PCM5100A y la conexión con sus contrapartes en el integrado LPC4337 de la EDU-CIAA-NXP.

TABLA 3.10: Interconexión de señales I²S entre PCM5100A y LPC4337.

PCM5100A	LPC4337	Descripción
DIN	I2S1_TX_SDA	Datos de audio digital PCM, un bit por reloj
LRCK	I2S1_TX_WS	Dato actual em DIN es del canal izquierdo o derecho
BCK	I2S1_TX_SCK	Reloj de los datos de audio
SCK	No disponible	Master Clock. Se genera internamente a partir de BCK

Existen restricciones sobre la generación interna del *Master Clock* a partir de la frecuencia de reloj de la señal de audio digital y su formato, no todos los formatos y frecuencias son soportadas [12, p.25]. Como se ve en la figura 3.31, la opción viable que demanda menos recursos de memoria es 16 bit estéreo a 32 kHz y por ese motivo será el formato de audio utilizado en este trabajo.

Sample f (kHz)	BCK (f_S)	
	32	64
8	–	–
16	–	1.024
32	1.024	2.048
44.1	1.4112	2.8224
48	1.536	3.072
96	3.072	6.144
192	6.144	12.288
384	12.288	24.576

FIGURA 3.31: Frecuencia «Sample f (kHz)» y bits por muestra estéreo «BCK (f_s)» necesarios para generar un *Master Clock* interno.

3.3.6. Mezclador de audio por software

El objetivo del mezclador es reproducir música y diferentes efectos de sonido al mismo tiempo.

Las actualizaciones del mezclador se hacen una vez por cuadro de video. En el modo de video definido en la subsección 3.2.7 hay aproximadamente 60 cuadros por segundo[15]. La duración de cada cuadro es de aproximadamente $16\,\mathrm{ms}$. A raiz de esto, el mezclador dispone de un área de memoria de trabajo suficiente para guardar el sonido que se reproduce en $16\,\mathrm{ms} + 1\,\mathrm{ms}$ para redondear hacia arriba y generar un margen de seguridad.

El proceso de mezclar y el de reproducir son distintos, en el primero el acceso a memoria es lectura/escritura por el software y en el segundo de solo lectura por el periférico I^2S del microcontrolador quien realiza la transmisión de esos datos hacia el PCM5100A.

Para evitar que se escriba sobre un área de memoria que se esta transmitiendo, se definen dos buffers iguales llamados A y B. En A se copia el fragmento de música y se suman los fragmentos de los efectos de sonido correspondientes al periodo actual mientras que B esta siendo transmitido. Al iniciar otro cuadro A y B intercambian sus datos y el proceso se repite. Esto introduce una latencia de un cuadro de video en el audio. El video, al utilizar una técnica de buffer doble similar, también sufre de la misma latencia. Por lo tanto, no se produce un desfase entre audio y video.

3.3.7. Salida de audio

La salida de audio desde el PCM5100A es de linea, analógica y no amplificada mediante las señales AUDIO_OUTL y AUDIO_OUTR. El puerto de salida de audio en RETRO-CIAA se denomina «AUDIO LINE OUT».

[15]Si bien el modo de video elegido es de $60\,\mathrm{Hz}$, los parámetros de la señal de video elegida en la sección 3.2.10 producen que efectivamente sea $59{,}86\,\mathrm{Hz}$.

3.4. Entradas de Joystick

El trabajo incluye dos entradas de joysticks o gamepads. Se decidió que fuesen compatibles con joysticks de «Family Game» debido a su disponibilidad y muy bajo costo en nuestro mercado. Se hace mención a joystick o gamepad de manera indistinta ya que a los fines de esta memoria solo importa la interfaz eléctrica y no la forma del dispositivo. De todos modos se aclaran ambos términos:

Gamepad También llamado *Joypad* o simplemente *Controller* o *Controlador*, es un dispositivo que se sostiene con ambas manos y la entrada de datos se realiza mayormente presionando botones con los pulgares. Sobre la izquierda se encuentran los controles de dirección (ir adelante, derecha, abajo o izquierda) y sobre el medio y la derecha se posiciona los botones de uso general. En la figura 3.32 se puede ver un ejemplo de gamepad.

FIGURA 3.32: Ejemplo de Gamepad.

Joystick Es un dispositivo cuya entrada de datos se realiza mediante la inclinación de una palanca con centro en dos ejes generalmente llamados X e Y. Los datos generados pueden ser analógicos (un desplazamiento proporcional sobre el eje X/Y) o digitales, en cuyo caso el movimiento de la palanca hacia adelante, derecha, abajo o izquierda produce el mismo resultado que la parte izquierda de un gamepad. Este dispositivo comúnmente tiene botones de uso general en la palanca o en su base. En la figura 3.33 puede verse un ejemplo de joystick.

FIGURA 3.33: Ejemplo de Joystick.

3.4.1. Señales

Las señales corresponden a las utilizadas por la consola de 8 bit Nintendo Famicom de la cual el «Family Game» es un clon.

El gamepad de «Family Game» está basado en un integrado CD4021B, esto es un *shift register* de 8 bit de la familia CMOS. En estos gamepads el integrado se utiliza como serializador de 8 entradas digitales correspondientes al estado de sus 8 botones. Posee ademas una entrada de reloj (CLK), una de latch (LATCH) y una salida serie (DATA). Cuando la señal de latch se vuelve positiva, se hace un muestreo de las 8 entradas digitales reteniendo sus valores en Flip-Flops tipo D y se presenta el estado de la primer entrada digital en la salida serie. En cada flanco ascendente del reloj se presenta en la salida serie el estado retenido de la siguiente entrada, en orden ascendente. Al 8vo ciclo de reloj se comienza otra vez, presentando el primer valor en la salida. Las señales del gamepad se resumen en la tabla 3.11.

TABLA 3.11: Señales de gampepad compatible con «Family Game».

Señal	Descripción
CLK	Reloj de corrimiento de datos a la salida
LATCH	Captura y retención de entradas digitales
DATA	Entrada digital a la salida según corrimiento
PWR	5 V de alimentación
GND	Retorno de la alimentación

El cable del gamepad esta directamente conectado a las entradas y salidas del integrado CD4021B y por esto eléctricamente todas las señales son CMOS. A pesar de no ser ideal, se decide alimentar al gamepad con 3,3 V ya que el microncontrolador de la EDU-CIAA-NXP y los buffers de entrada y salida funcionan a 3,3 V.

Esto es posible ya que el CD4021B soporta alimentación a partir de 3 V y CMOS define un valor alto también a partir de 3 V. Se tiene en cuenta que debido a la baja tensión de alimentación es necesario reducir la velocidad de operación aumentando las demoras especificadas en la hoja de datos. En consecuencia, se utiliza una demora luego de cada cambio de nivel de LATCH o CLK especificada en la ecuación 3.8.

$$JCLK_{tWidth} = 200\,\text{ns} \tag{3.8}$$

Cada entrada en el *shift register* esta conectado a un botón especifico del gamepad. En la siguiente secuencia se enumera la operación que captura cada uno de los 8 bits de datos y entre paréntesis se indica el botón correspondiente: Latch (A), Clock 1 (B), Clock 2 (Select), Clock 3 (Start), Clock 4 (Up), Clock 5 (Down), Clock 6 (Left) y Clock 7 (Right).

3.4.2. Conectores

En RETRO-CIAA se pueden conectar hasta dos joysticks de «Family Game» en los dos puertos D-sub macho de 9 posiciones dispuestos para tal fin y marcados como

«GAMEPAD 1» y «GAMEPAD 2». En la figura 3.34 se observa la numeración de los pines del puerto y en la tabla 3.12 una lista con la disposición de las señales.

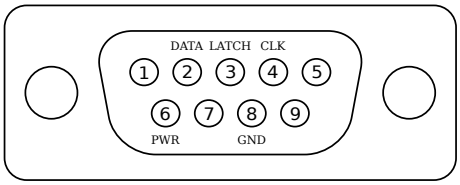


FIGURA 3.34: Numeración de pines y señales en los puertos de joystick.

TABLA 3.12: Disposición de señales en los puertos de joystick.

Posición	Señal	Posición	Señal	Posición	Señal
1	–	4	CLK	7	–
2	DATA	5	–	8	GND
3	LATCH	6	PWR	9	–

En la tabla 3.13 se lista la conexión de cada señal entre pines específicos de los puertos de joystick «GAMEPAD 1» y «GAMEPAD 2» y el pin de expansión de la EDU-CIAA-NXP. También se indica si el pin de expansión se configura como Entrada o Salida desde el microcontrolador. La alimentación de $3{,}3\,\mathrm{V}$ y el retorno de los joysticks se conectan a la alimentación y retorno comunes de la placa.

TABLA 3.13: Conexión de señales entre pines en puertos de joystick y la EDU-CIAA-NXP.

Señal	GP1	GP2	EDU-CIAA-NXP	E/S	Comentario
JOY1_DATA	2		GPIO5	E	Datos de joystick 1
JOY2_DATA		2	GPIO7	E	Datos de joystick 2
JOY_LATCH	3	3	GPIO1	S	Latch para ambos joysticks
JOY_CLK	4	4	GPIO6	S	Reloj para ambos joysticks

Los joysticks de igual conector pero de norma ATARI como los de Commodore 64, MSX, Commodore AMIGA, SEGA Master System o SEGA Mega Drive[16]no son compatibles con este trabajo.

3.4.3. Operación

Se define que la lectura de los joysticks se actualiza una vez por cuadro de video, específicamente en la primer linea del *Vertical Blanking Interval* y desde el driver de video ejecutado por el núcleo M0.

$JDATA_{tRead}$ es el tiempo aproximado de lectura de JOY1_DATA y JOY2_DATA y a los fines de diseño se estiman 16 MCU_{cucle} en la ecuación 3.9. El tiempo total de lectura de joysticks se calcula en la ecuación 3.10.

$$JDATA_{tRead} = MCU_{cycle} \cdot 16 = 78{,}43\,\mathrm{ns} \tag{3.9}$$

[16]Consola de 16 bit de SEGA. También llamada SEGA Genesis en algunos mercados. Aquí en el país tuvo mucho éxito como sucesora del «Family Game» y se la conoció simplemente como «Sega».

$$JSTAT_{tGet} = JCLK_{tWidth} \cdot 16 + JDATA_{tRead} \cdot 8 \tag{3.10}$$
$$= 200\,\text{ns} \cdot 16 + 78{,}43\,\text{ns} \cdot 8 = 3{,}827\,44\,\mu\text{s}$$

El tiempo de lectura de joysticks ($JSTAT_{tGet}$) no debe ser mayor al definido en la sección 3.2.10 para el tiempo de linea visible (*vdisp*) o hasta un máximo del tiempo de linea total (*vtotal*) si la implementación del driver de video lo permite.

En la figura 3.35 puede verse el estado de las señales y su temporizado en el proceso de lectura de los joysticks.

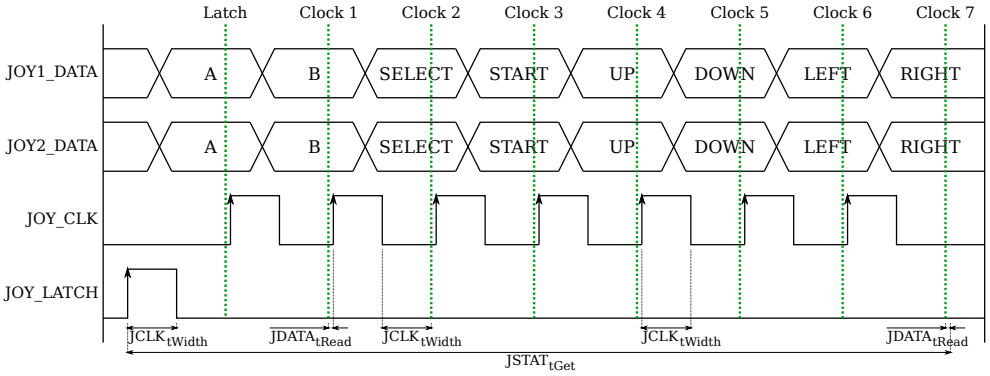


FIGURA 3.35: Señales y temporizado de lectura de joysticks. En verde (linea punteada) se marcan las lecturas de cada botón.

3.5. Wi-Fi / Bluetooth

Se suma conectividad inalámbrica Wi-Fi y Bluetooth Low Energy (BLE) mediante un módulo ESP-WROOM-32 de la empresa Espressif Inc. como se muestra en la figura 3.36. Este módulo se opera a través de una interfaz serie tipo UART.

FIGURA 3.36: Modulo Wi-Fi / Bluetooth ESP-WROOM-32.

3.5.1. Restricciones

Se encontraron las siguientes restricciones para utilizar el módulo Wi-Fi:

- El único puerto serie disponible en los pines de expansión de la EDU-CIAA-NXP es UART3 y las señales CTS y RTS no están disponibles.

- El manual de hardware del módulo recomienda reservar para uso propio unos 500 mA en la fuente de 3,3 V [11, p.4]. El regulador de tensión de 3,3 V de la EDU-CIAA-NXP podría no ser suficiente.

3.5.2. Señales

En la tabla 3.14 se listan las conexiones entre el módulo Wi-Fi y los pines de expansión de la EDU-CIAA-NXP, indicándose si estas ultimas se configuran como entrada o salida. Básicamente las señales RXD y TXD del módulo se conectan al puerto serie UART3 del microcontrolador de la EDU-CIAA-NXP y las señales CTS, RTS y EN se conectan a pines GPIO.

TABLA 3.14: Conexión de señales entre el módulo Wi-Fi y la EDU-CIAA-NXP.

Señal	Modulo Wi-Fi	EDU-CIAA-NXP	E/S	Comentario
WM_AT_RXD	IO16	232_TX	S	(UART3) Comandos AT
WM_AT_TXD	IO17	232_RX	E	(UART3) Comandos AT
WM_AT_RTS	IO14	CAN_RD	E	RTS por GPIO
WM_AT_CTS	IO15	CAN_TD	S	CTS por GPIO
WM_EN	EN	TEC_C1	S	Habilitación de módulo

Existen señales propias del módulo para debug y configuración que se exponen como tira de pines o slots en la RETRO-CIAA. La tabla 3.15 lista estas señales y su función.

TABLA 3.15: Señales propias del módulo Wi-Fi para debug y configuración.

Señal	Modulo Wi-Fi	E/S	RETRO-CIAA	Comentario
WM_DEBUG_TX	TXD0	S	WIFI UART TD	Debug log
WM_DEBUG_RX	RXD0	E	WIFI UART RD	
WM_BOOT_OPT	IO0	E	WIFI BOOT	Actualización de firmware

En cuanto se activa el módulo Wi-Fi su puerto de debug envia datos de inicio por la señal WM_DEBUG_TX. La configuración necesaria para leerlos se lista en la tabla 3.16.

TABLA 3.16: Configuración de la UART para recibir datos de debug del módulo Wi-Fi.

Parametro	Valor
Baud rate	115200
Data bits	8
Stop bits	1
Parity	None
Flow control	XON/XOFF

3.5.3. Actualización de firmware por UART

El pin de la señal WM_BOOT_OPT se coloca junto a un pin de masa en la RETRO-CIAA. A ese conjunto de dos contactos se los denomina «WIFI BOOT». Si al

momento del arranque del módulo los contactos se mantienen abiertos, se ejecutará el código del firmware grabado en su memoria interna SPI. Si en cambio los contactos están cortocircuitados con un jumper (WM_BOOT_OPT llevado a masa) entonces el módulo se pondrá en modo de actualización de firmware por UART. Para llevar a cabo la actualización es necesario conectar las señales WM_-DEBUG_TX y WM_DEBUG_RX con un adaptador USB-TTL a la PC y bajar la herramienta «Flash Download Tool» desde el sitio web del fabricante.

3.5.4. Operación

Primero es necesario activar el módulo Wi-Fi asignando un valor alto a la señal WM_EN. Una vez iniciado, el mismo se comanda a través de la UART3 del microcontrolador usando comandos AT del fabricante [10]. A modo de ejemplo, a continuación se listan algunos comandos útiles para este trabajo.

AT+RST Reinicia el módulo.

AT+GMR Información de versión del firmware.

AT+UART_CUR Configuración actual de UART, no se guarda en FLASH. Consulta actual con «AT+UART_CUR?». «AT+UART_CUR=115200,8,1,0,0» asigna 115200 8n1 sin flow control.

AT+CWMODE Asigna el modo Wi-Fi del adaptador. Consulta modo actual con «AT+CWMODE?». «AT+CWMODE=1» setea modo «Station».

AT+CWJAP Conecta a un *Punto de acceso Wi-Fi*. Consulta actual con «AT+CWJAP?». «AT+CWJAP=“RouterWifi”,“1234”» conecta a «RouterWifi» usando la contraseña «1234».

AT+CWLAP Lista los *Puntos de acceso Wi-Fi* disponibles.

AT+CIPSTART Inicia conexion TCP, transmision UDP o conexion SSL. Por ejemplo «AT+CIPSTART=“TCP”,“iot.espressif.cn”,8000».

AT+CIPSEND Envia datos a traves de la conexion establecida. Por ejemplo «AT+CIPSEND=32» envía 32 B. El tamaño no puede ser mayor a 2048 B.

3.6. Almacenamiento

Se incluye almacenamiento no volátil mediante una EEPROM I^2C M24M01 de STMicroelectronics N.V. en encapsulado TSSOP8 como se ve en la figura 3.37. Se dispone de 1 Mbit (128 kB) dividido en páginas de 256 B. Este dispositivo es compatible con los modos I^2C de 100 kHz (Standard), 400 kHz (Fast) y 1 MHz (Fast Mode Plus).

FIGURA 3.37: Memoria EEPROM M24M01 en encapsulado TSSOP8.

3.6.1. Acceso a memoria

Una memoria de 128 kB necesita 17 bits de direccionamiento ($\frac{2^{17}}{1024} = 128$).

En este caso se accede a la memoria del integrado M24M01 mediante dos direcciones I^2C como si fuesen dos dispositivos distintos de 64 kB cada uno. Una vez elegido el segmento de 64 kB sobre el cual se quiere trabajar, se envía a esa dirección I^2C dos bytes indicando el lugar exacto en memoria al que se quiere acceder dentro del segmento [20, p.13].

En la tabla 3.17 se listan las direcciones I^2C del dispositivo.

TABLA 3.17: Direcciones I^2C de la EEPROM M24M01.

Dirección I^2C	Memoria disponible
1010110	64 kB
1010111	64 kB

3.7. Buffers de E/S

Se decidió proteger los pines de E/S de la EDU-CIAA-NXP con buffers SN74ALVC245 de Texas Instruments. Se eligió este integrado ya que cumple la misma función en el módulo de salida VGA «PmodVGA» de Digilent Inc. El encapsulado es TSSOP20 y puede verse en la figura 3.38.

FIGURA 3.38: Buffer SN74ALVC245 en encapsulado TSSOP20.

Se utilizan dos buffers denominados «VGA SYNC/JOY BUFFER» y «VGA RGB BUFFER» según las señales a proteger. En la tablas 3.18 y 3.19 se listan las señales de entrada y salida de cada buffer. Las direcciones de entrada y salida (pin DIR en el integrado) se configuran como «de puerto A hacia puerto B».

TABLA 3.18: Señales en buffer «VGA SYNC/JOY BUFFER».

Entrada (A)	Salida (B)	Origen	Destino
VSYNC	VSYNC_OUT	EDU-CIAA-NXP	VGA Port
$\overline{\text{HSYNC}}$	$\overline{\text{HSYNC\_OUT}}$	EDU-CIAA-NXP	VGA Port
JOY_LATCH	JOY_LATCH_OUT	EDU-CIAA-NXP	GAMEPAD Ports
JOY_CLK	JOY_CLK_OUT	EDU-CIAA-NXP	GAMEPAD Ports
JOY1_DATA	JOY1_DATA_IN	GAMEPAD Port 1	EDU-CIAA-NXP
JOY2_DATA	JOY2_DATA_IN	GAMEPAD Port 2	EDU-CIAA-NXP
LED_Y	LED_Y_OUT	EDU-CIAA-NXP	Backlight
LED_Z	LED_Z_OUT	EDU-CIAA-NXP	Backlight

Los buffers poseen una señal de activación «$\overline{\text{OE}}$». En la tabla 3.20 se listan las señales y el pin que las maneja del puerto de expansión de la EDU-CIAA-NXP.

TABLA 3.19: Señales en buffer «VGA RGB BUFFER». RED, GREEN y BLUE son señales analógicas producto de la combinación de sus componentes binarios a través de divisores de tensión.

Entrada (A)	Salida (B)	Origen	Destino
R2	RED	EDU-CIAA-NXP	VGA Port
R1	RED	EDU-CIAA-NXP	VGA Port
R0	RED	EDU-CIAA-NXP	VGA Port
G2	GREEN	EDU-CIAA-NXP	VGA Port
G1	GREEN	EDU-CIAA-NXP	VGA Port
G0	GREEN	EDU-CIAA-NXP	VGA Port
B1	BLUE	EDU-CIAA-NXP	VGA Port
B0	BLUE	EDU-CIAA-NXP	VGA Port

TABLA 3.20: Señales de activación ($\overline{OE}$) de los buffers.

Señal	EDU-CIAA-NXP
$\overline{SYNC\_JOY\_OE}$	LCD_EN
$\overline{RGB\_OE}$	TEC_C0

3.8. Esquemático

Se utilizó la aplicación Eeschema del KiCAD 4. El diseño se llevó a cabo conectando los puertos de expansión -o interfaz- de la EDU-CIAA-NXP a los diferentes subsistemas y en hojas jerárquicas individuales para facilitar su comprensión. Se crearon los componentes originales necesarios en la librería «retro-ciaa.lib» y finalmente se corrió el chequeo de reglas eléctricas (ERC).

El esquemático posee 8 hojas. La hoja 1/8 es la vista jerárquica de los subsistemas y las siguientes 7 hojas contienen un subsistema cada una. En la figura 3.39 se observa dicha estructura.

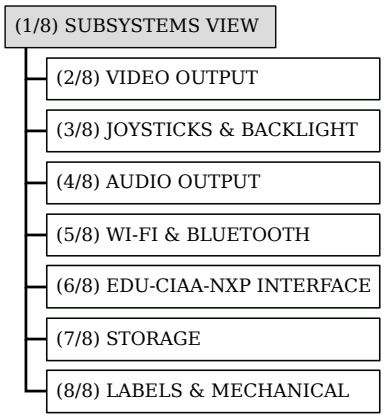


FIGURA 3.39: Estructura de hojas del esquemático.

El esquemático se incluye en el apéndice A.

3.9. PCB

Luego de asignar huellas de componentes con CvPcb, se exportó el archivo «netlist» con las conexiones del esquemático diseñado en la sección 3.8 y se diseñó un PCB de doble faz en la aplicación Pcbnew de KiCAD 4. Las tareas de diseño se concentraron en las siguientes actividades:

- Creación de huellas de componentes inexistentes en la librería de KiCad.
- Ubicación de la huella de las dos tiras de pines para calzar en los puertos de expansión P1 y P2 de la EDU-CIAA-NXP.
- Definición del área de la placa, tomando en cuenta recortes necesarios para no bloquear funcionalidad existente como el acceso a pulsadores, conectores y visibilidad de leds.
- Ubicación de los componentes en la placa.
- Ruteo de todas las pistas.
- Ubicación de los planos de masa.
- Arreglo de la serigrafía.
- Chequeo de reglas de diseño (DRC).

Se toma la serigrafía de la EDU-CIAA-NXP para orientarse en la superficie de la placa.

A continuación se comentan los criterios de diseño, uno en cada subsección.

3.9.1. Ancho de pistas y vías

El ancho de pista para señales se fijo en $0{,}25\,\mathrm{mm}$ con un espaciado de $0{,}2\,\mathrm{mm}$. El diámetro de vía es de $0{,}6\,\mathrm{mm}$ con tamaño de taladro es de $0{,}4\,\mathrm{mm}$.

Para potencia se utiliza un ancho de pista de $0{,}75\,\mathrm{mm}$ o $1\,\mathrm{mm}$ según los requisitos de corriente y diámetro de vía de $0{,}8\,\mathrm{mm}$ con taladro $0{,}6\,\mathrm{mm}$.

3.9.2. Definición del área de la placa

Las dimensiones de la placa son iguales a la EDU-CIAA-NXP con excepción del borde inferior que fue extendido $8\,\mathrm{mm}$ para alojar los conectores de joysticks sin bloquear los pulsadores de la EDU-CIAA-NXP. Se agregan 4 agujeros de montaje que coinciden con la placa base. El área total del PCB es de $145\,\mathrm{mm}$ de alto por $86\,\mathrm{mm}$ de ancho.

3.9.3. Ubicación de componentes

Los componentes se ubicaron tomando en cuenta las siguientes consideraciones:

- Las restricciones de espacio, con las conexiones a los puertos de expansión P1 y P2 bloqueando el uso de los costados izquierdo y derecho de la placa.

- La comodidad del usuario, por lo que se decide colocar los puertos de E/S en los extremos superior e inferior.
- No bloquear la conectividad existente en la EDU-CIAA-NXP.

El DAC de audio se posicionó del lado del puerto P2 que es donde se encuentran las señales I^2S y lo mas cerca que fue practico al conector de audio en el extremo superior izquierdo de la placa a fin de intentar minimizar el largo de sus señales analógicas.

Los buffers concentran una veintena de señales proveniente de los puertos de expansión. Por este motivo se situaron a corta distancia de estos puertos.

Los integrados I^2C de EEPROM y buffer E-DDC se sitúan cerca del origen de esas señales en el puerto de expansión P1.

El módulo Wi-Fi debe estar lo mas alejado posible de otros componentes ya que puede generar o recibir interferencia. Por recomendación del fabricante, la antena PCB del módulo debe sobresalir del PCB en donde se monta [11].

El conector VGA se ubica al centro del extremo superior de la placa y sus resistencias lo mas cerca posible de los pines de señal.

Los dos puertos de joystick se ubican en el extremo inferior y en dirección al usuario.

En la figura 3.40 se observa el resultado de ubicar los componentes intentando cumplir lo mejor posible con estas consignas y restricciones.

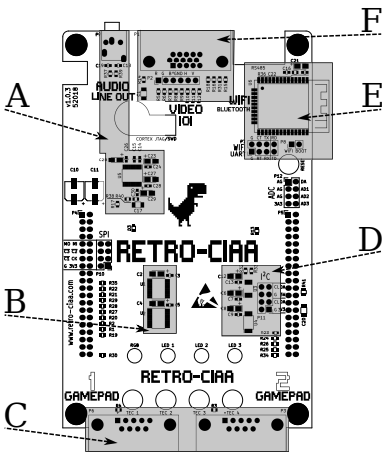


FIGURA 3.40: Ubicación de componentes en RETRO-CIAA. (A) DAC de audio y salida de linea analógica. (B) Buffers. (C) Entradas de joystick. (D) EEPROM y buffer E-DDC. (E) Modulo Wi-Fi. (F) Salida de video VGA.

3.9.4. Ruteo de pistas

El método de ruteo utilizado para señales (no alimentación) consiste en desarrollar las pistas en horizontal en la capa de cobre del frente y en vertical en la capa de cobre del dorso. Una ventaja de este método es que se aprovecha muy bien el espacio. Ademas, reduce el acople capacitivo entre señales en diferentes planos

ya que las pistas siempre se cruzan a 90 grados. La desventaja es que cada cambio de dirección implica una vía para salir hacia la otra capa.

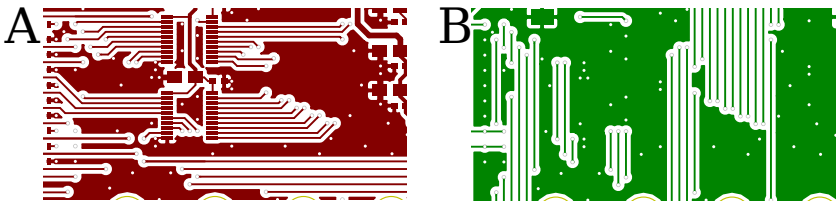


FIGURA 3.41: Ruteo de pistas en PCB de RETRO-CIAA. (A) Pistas horizontales en capa F.Cu. (B) Pistas verticales en capa B.Cu.

3.9.5. Planos de masa

En las capas de frente y dorso se define un plano de masa digital conectado a los pines de GND de la EDU-CIAA-NXP que cubre casi toda la placa y tres islas que cumplen requerimientos específicos de componentes en sus sectores correspondientes:

- Isla que cubre la mitad analógica del DAC de audio y la salida de linea analógica, conectada a los pines de GNDA de la EDU-CIAA-NXP.
- Isla que cubre la mitad digital del DAC de audio, conectada al plano de masa digital de la placa en un solo punto.
- El área que cubre el módulo Wi-Fi, conectada al plano de masa digital de la placa en un solo punto.

En la figura 3.42 puede verse la ubicación de estas islas.

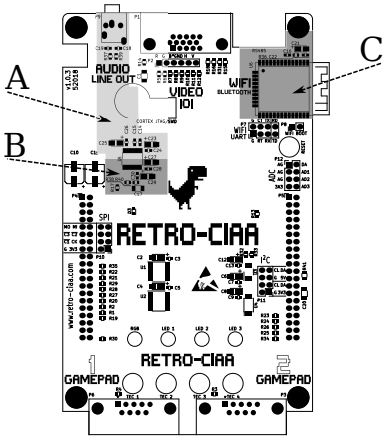


FIGURA 3.42: Ubicación de islas de plano de masa en RETRO-CIAA. (A) Mitad analógica del DAC de audio y salida de linea analógica. (B) Mitad digital del DAC de audio. (C) Área cubierta por módulo Wi-Fi.

3.9.6. Serigrafia

Se muestra claramente la anotación de cada componente, la polarización de capacitores, el ánodo/cátodo en diodos y el nombre y disposición de señales en

los slots y tiras de pines. Se marcan algunos componentes con un estilo «retro». Para esto se convirtieron imágenes a huellas de serigrafía con la aplicación Bitmap2Component de KiCad 4. En la figura 3.43 se resalta la serigrafía de parte de la placa.

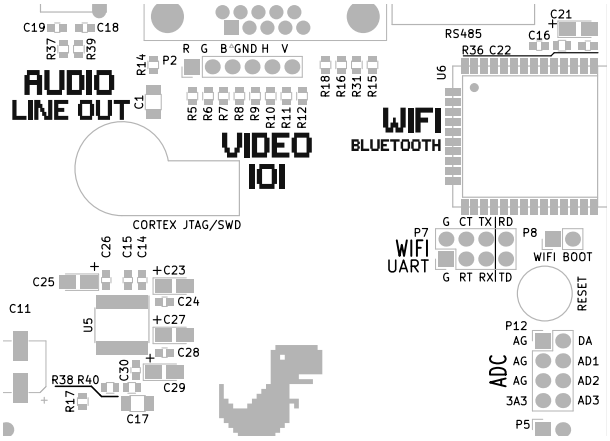


FIGURA 3.43: Ejemplo de serigrafía de la placa RETRO-CIAA.

3.9.7. Backlight

Se retroilumina el logo y el nombre del trabajo a través del sustrato traslucido del PCB. Para esto fue necesario crear dos huellas de Pcbnew que dejan libre de cobre y de mascara antisoldante las áreas que no bloquean la luz. En la figura 3.44 se muestra el logo y nombre que se retroilumina.

FIGURA 3.44: Logo y nombre retroiluminado.

3.10. Firmware

Se comentan decisiones de diseño que afectan el desarrollo del Firmware.

3.10.1. Herramientas

Las herramientas utilizadas para desarrollo son las siguientes:

- Linux, distribución Debian 9.
- Lenguaje de programación 'C' y Assembler.

- Compilador GCC, toolchain arm-eabi.
- Makefiles para la compilación del código.
- Git para versionado de código.
- Programación de flash y debug por integrado FTDI de la EDU-CIAA-NXP.

3.10.2. Normas de codificación

- No utilizar typedef para ofuscar struct o enum.
- Evitar la múltiple inclusión de headers con «#pragma once».
- Ningún header 'C' contiene código de soporte para compilación desde 'C++'. Esto debe implementarse incluyendo los headers 'C' desde un header de 'C++' (.hh) y aplicando allí el soporte necesario.
- Utilizar tipos de dato básicos definidos en stdint.h y stdbool.h.
- Siempre aplicar la opción de compilación -Wall y resolver todos los warnings.
- Programación defensiva: corroborar que todo parámetro recibido en una función este entre los limites de lo esperado o salir devolviendo un error.
- Enfoque modular. Un módulo por cada funcionalidad.
- Anteponer el prefijo «g_» a toda variable global, aun si no es visible para otros archivos de código.
- Reducir al mínimo el tamaño y la complejidad.
- Reutilizar funcionalidad.

3.10.3. Módulos y convención de nombres

El código se divide en módulos que aportan funcionalidad especifica al firmware.

Cada módulo tiene un nombre en minúsculas, simple, corto, descriptivo y en lo posible de una sola palabra. De ser dos o mas se usa guion bajo como separador.

Esta contenido en un archivo cuyo nombre es el nombre del módulo, una estructura principal (struct) con nombre de tipo de dato igual pero en mayúsculas. El nombre en mayúsculas mas guion bajo actuá como prefijo de todas las funciones y constantes exportables de ese módulo.

3.10.4. Almacenamiento

El microcontrolador LPC4337JBD144 dispone de dos bancos de memoria Flash con un puerto de acceso independiente para cada uno [23, p.64].

El acceso de los dos núcleos a un mismo banco de memoria resulta en tiempos de acceso no deterministas o no ideales, especialmente cuando ambos están leyendo instrucciones de programa y datos desde el mismo banco de memoria. El núcleo M0 es especialmente sensible ya que debe mantener una velocidad de ejecución

determinista aun a costo de desaprovechar almacenamiento. Por lo tanto se definió que cada banco de memoria Flash se asigne exclusivamente a un núcleo.

En la tabla 3.21 se listan las direcciones de los dos bancos de memoria, el espacio disponible en cada uno y a que núcleo fue asignado.

TABLA 3.21: Bancos de memoria Flash asignados a cada núcleo.

Banco de Flash	**Dirección base**	**Tamaño**	**Núcleo asignado**
A	1A 00 00 00 h	512 KiB	Cortex-M4
B	1B 00 00 00 h	512 KiB	Cortex-M0

Capítulo 4

Implementación

Se divide la implementación en dos secciones: Hardware y Firmware.

4.1. Hardware

Para la implementación del hardware fue necesario cumplir con los siguientes pasos:

- Considerar particularidades del fabricante de PCB.
- Exportar los archivos de diseño del PCB para entregar a fabrica.
- Fabricación del PCB.
- Exportar el BOM y pedido de componentes.
- Montaje de 5 prototipos.
- Pruebas de continuidad y alimentación.

Se detallan cada uno de estos pasos en la subsección correspondiente.

4.1.1. Considerar procesos del fabricante del PCB

El diseño del PCB no sufrió limitaciones en el proceso de fabricación ya que los parámetros utilizados coinciden con el proceso mas económico de un fabricante internacional.

No obstante, fue necesario tener en cuenta que en general los fabricantes tienen una precisión de aproximadamente $0{,}05\,\mathrm{mm}$ en la alineación de la mascara antisoldante con la cara de cobre. Por ese motivo a toda área de *ausencia de* mascara antisoldante le aplican compulsivamente una expansión del $5\,\%$. El objetivo es que el margen de error de la alineación no produzca que se fije mascara sobre parte del área de cobre de los pads.

En este trabajo se hace un uso no convencional del sustrato del PCB para retroiluminar un dibujo y se necesita que ese dibujo realizado en la mascara antisoldante no sufra absolutamente ninguna modificación o ajuste automático. Si bien una expansión de $5\,\%$ en la mascara de un pad de $0{,}25\,\mathrm{mm}$ es imperceptible, a un dibujo de $50\,\mathrm{mm}$ lo arruina por completo.

Por este motivo antes de exportar el diseño para su fabricación, se aplicó manualmente un $5\,\%$ de expansión de mascara antisoldante en los pads y se habló con la

fabrica a fin de solicitar que no realicen ningún ajuste o expansión automático de la mascara antisoldante.

4.1.2. Archivos de diseño para fabrica

La fabrica solicita archivos de diseño en formato Gerber. Estos archivos se exportaron desde Pcbnew de KiCad 4 utilizando la opción de menú File->Plot. Las opciones pueden verse en la figura 4.1. Se exportaron las dos caras de cobre F.Cu y B.Cu, el área de pasta de soldar de la cara superior F.Paste para fabricación del stencil, la mascara antisoldante -F.Mask, B.Mask- y serigrafía de ambas caras -F.Silk, B.Silk- y el contorno de la placa en Edge.Cuts.

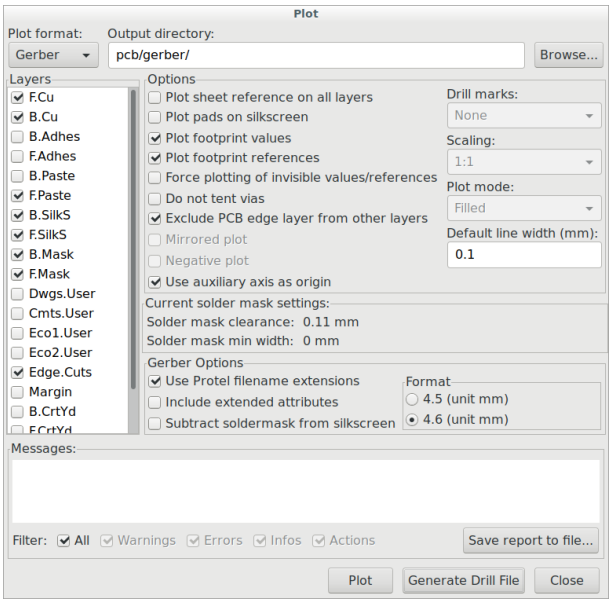


FIGURA 4.1: Opciones usadas para exportar archivos Gerber.

Los archivos de taladro se exportaron usando la opción File->Plot->[Generate Drill File]. Las opciones usadas pueden verse en la figura 4.2.

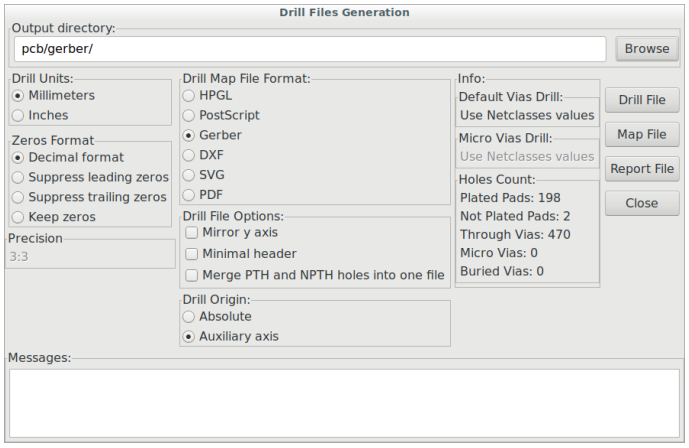


FIGURA 4.2: Opciones usadas para exportar archivos de taladro.

4.1.3. Fabricación del PCB

Se utilizaron los servicios de Elecrow en China. Se encargaron 25 unidades con las características listadas en la tabla 4.1.

TABLA 4.1: Características técnicas del PCB de RETRO-CIAA.

Parametro	Valor
Dimensiones	86 mm×145 mm
Material	FR-4
Espesor	1,6 mm
Capas	2
Terminación	HASL
Mascara antisoldante	Negro mate
Serigrafia	Blanca, en ambas caras

En la figura 4.3 puede verse una de las placas fabricadas.

FIGURA 4.3: Foto del PCB fabricado. La imagen del T-Rex pertenece al proyecto Chromium, autor Sebastien Gabriel, licencia BSD 3-clause.

4.1.4. Pedido de componentes

Se compraron componentes para diez prototipos. Los componentes se encargaron directamente al proveedor Mouser Electronics de EEUU.

Se debió abonar 35 USD de envió mas impuestos por la posición arancelaria de cada componente, esto es derechos de importación mas IVA según el tipo de producto.

4.1.5. Lista de materiales

En la tabla 4.2 se lista un resumen con el numero de parte y la descripción de los conectores y circuitos integrados utilizados. La lista no incluye capacitores ni resistencias.

TABLA 4.2: Conectores y circuitos integrados de la RETRO-CIAA.

Número de parte	Descripción
PCM5100A	DAC de audio estéreo de 100 dB
SN74ALVC245	Buffer de 8 puertos
TCA9617B	Buffer I^2C
M24M01	EEPROM I^2C de 1 Mbit
ESP-WROOM-32	Módulo Wi-Fi y Bluetooth
SJ-2509N	Conector de audio Jack de 2,5 mm
ID09P33E4GX00LF	Conector DB-9 macho THT
ICD15S13E4GV00LF	Conector DB-15HD hembra THT
LW Y1SG	LED luz blanca de emisión lateral
TE_2118731-2	Backlight cover

En el apéndice B se incluye el BOM (*Bill Of Materials*) completo.

4.1.6. Montaje de prototipos

Se montaron a mano cinco prototipos. Se coloco pasta de soldar con stencil (figura 4.4) y se ubicó cada componente en su posición en el PCB. Luego se soldó con aire caliente. En la figura 4.5 pueden verse los prototipos montados.

FIGURA 4.4: Pasta de soldar aplicada con stencil en huella de 0603 pulgadas.

4.1.7. Pruebas de continuidad y alimentación

Se probó con un multimetro la resistencia entre diferentes puntos de masa y alimentación de la RETRO-CIAA. Luego se armó un banco de pruebas. Se conectó a masa, se aplicaron 5 V y 3,3 V con una fuente de laboratorio, se observó el consumo y con un multimetro se miden tensiones en diferentes puntos de la placa.

Se conectaron las señales de los puertos de expansión con cables de tipo «Dupont», activando los buffers de E/S e inyectando señales que se replicaron del otro lado del buffer. De este modo se probaron los diferentes niveles de tensión

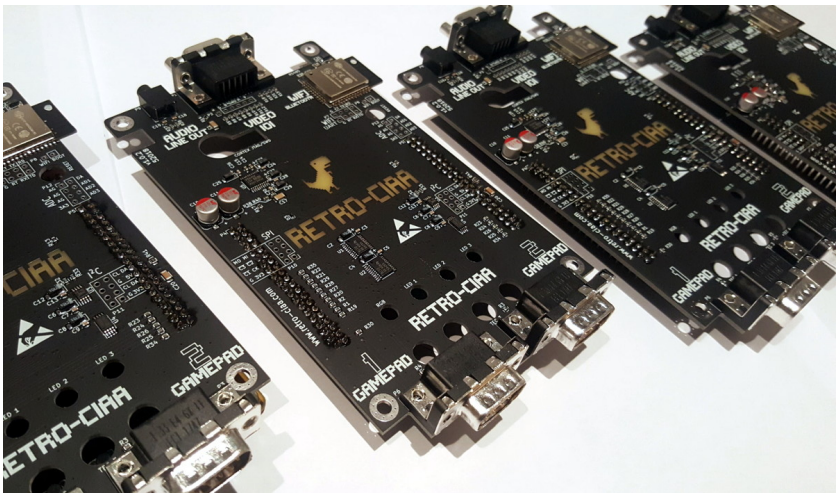


FIGURA 4.5: Prototipos de RETRO-CIAA.

analógica de los componentes del puerto VGA, corroborando sus niveles según el peso de los bits de color.

Finalmente se activó y se comprobó que el modulo Wi-Fi estaba vivo mediante la recepción de sus mensajes de inicio por el puerto serie de debug de ese componente.

Comprobado que no hay cortos y la alimentación y señales funcionaron como se esperaba, se desarmó el banco de pruebas y se conectó la placa RETRO-CIAA a la EDU-CIAA-NXP. Se comprobó que las lecturas de tensión en ambas eran normales, lo que dio inicio a la programación del firmware.

En la figura 4.6 se observa la placa de expansión RETRO-CIAA instalada sobre la EDU-CIAA-NXP.

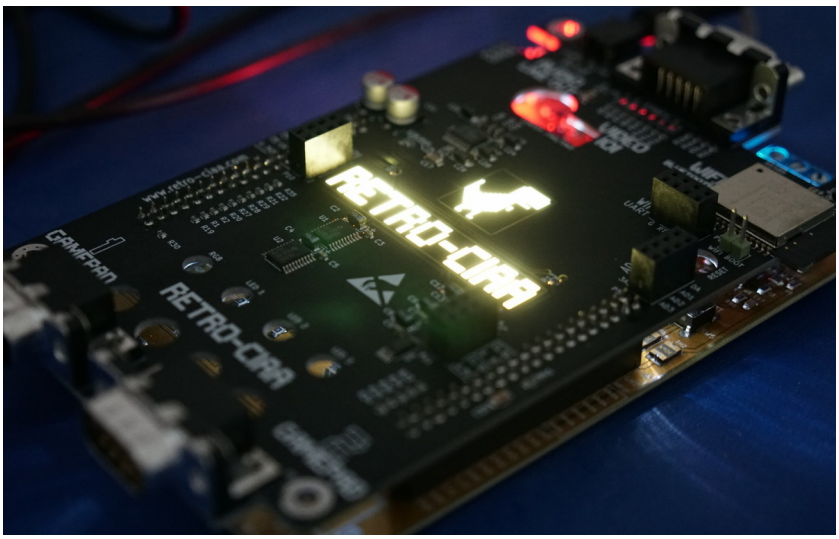


FIGURA 4.6: RETRO-CIAA instalada sobre EDU-CIAA-NXP.

4.2. Firmware

Una vez implementado el hardware se iniciaron los trabajos en el firmware de acuerdo a las pautas de diseño delineadas en el capitulo 3 que guiaron el desarrollo de las siguientes funcionalidades:

- Inicialización del hardware.
- Driver de video.
- Funciones gráficas.
- Funciones de audio.
- Aplicaciones de prueba.

4.2.1. Herramientas utilizadas

El desarrollo de software se realizó sobre el sistema operativo Debian 9. Para la edición de código fuente, el lanzamiento de comandos de build y la programación de la memoria flash del microcontrolador se utilizó Qt Creator 4.20 con plugin BareMetal.

4.2.2. Librería del fabricante

El fabricante del microcontrolador de la EDU-CIAA-NXP brinda de manera gratuita librerías en lenguaje 'C' para facilitar la configuración y el uso de periféricos y características.

El firmware de la RETRO-CIAA incluye la librería LPCOpen versión 3.02.

La librería incluye código de soporte para iniciar y configurar placas enteras en lo que se conoce como «board». Este trabajo agrega soporte (inconcluso al momento de escribir esta memoria) para una nueva placa llamada «lpcopen_edu_retro-ciaa_board».

4.2.3. Inicialización del hardware

En la RETRO-CIAA no hay hardware que deba iniciarse, de modo que las tareas de inicialización se centran en configurar los pines de entrada y salida del microcontrolador de la EDU-CIAA-NXP.

La mayoría de pines[1]cuentan con un multiplexor denominado SCU (*System Control Unit*) que sirve para elegir la opción de cual periférico asume el control de ese pin y que señal representa. Cada pin tiene una lista predefinida de hasta 8 señales de periféricos para elegir [22, p.84].

Existen periféricos de GPIO, de UART, EMC (puerto de memoria externa), USB, SSP (SPI) entre otros. Una vez asignada la señal del periférico al pin, solo resta realizar la configuración particular para ese periférico. Por ejemplo en el caso de

[1]Generalmente los pines sin función configurable son analógicos para uso dedicado del DAC, ADC o de buses como I^2C o USB.

un GPIO, esencialmente se debe indicar si el pin cumplirá el rol de entrada o salida.

Hay periféricos que demandan varios pines para exponer todas sus señales. Por ejemplo para utilizar un puerto UART completo, mediante el SCU se necesita asignar 4 señales a pines: TXD, RXD, CTS y RTS. Luego se configura dicho puerto para indicar la velocidad de operación de los pines de TXD y RXD, los bits de inicio, datos, parada y activar el control de flujo por hardware a través los pines CTS (*Clear to Send*) y RTS (*Request to Send*).

Se utilizan extensamente funciones de la librería LPCOpen identificadas con prefijo Chip_SCU, Chip_GPIO, Chip_I2S, NVIC_, etc.

Video

En el listado 4.1 se inician los pines de salidas digitales para los componentes de color, señales de sincronismo y habilitación del buffer para E-DDC.

El puerto E-DDC esta unido al bus I^2C de todo el sistema manejado por los pines dedicados I2C0_SCL e I2C0_SDA. El pin de la señal EDDC_EN no se configura como FAST ya que no se requiere que su estado cambie a alta velocidad.

LISTADO 4.1: Código de incialización de pines de video.

```
#define BOARD_I2C_MODE          I2C0_STANDARD_FAST_MODE
#define BOARD_I2C_SPEED         100000

// Configuración de SCU para los pines de video
Chip_SCU_PinMuxSet (6, 12, SCU_PINIO_FAST | SCU_MODE_FUNC0);    // GPIO8:  R2
Chip_SCU_PinMuxSet (4,  6, SCU_PINIO_FAST | SCU_MODE_FUNC0);    // LCD3:   R1
Chip_SCU_PinMuxSet (4,  5, SCU_PINIO_FAST | SCU_MODE_FUNC0);    // LCD2:   R0
Chip_SCU_PinMuxSet (4,  4, SCU_PINIO_FAST | SCU_MODE_FUNC0);    // LCD1:   G2
Chip_SCU_PinMuxSet (4,  3, SCU_PINIO_FAST | SCU_MODE_FUNC0);    // T_FIL3: G1
Chip_SCU_PinMuxSet (4,  2, SCU_PINIO_FAST | SCU_MODE_FUNC0);    // T_FIL2: G0
Chip_SCU_PinMuxSet (4,  1, SCU_PINIO_FAST | SCU_MODE_FUNC0);    // T_FIL1: B1
Chip_SCU_PinMuxSet (4,  0, SCU_PINIO_FAST | SCU_MODE_FUNC0);    // T_FIL0: B0
Chip_SCU_PinMuxSet (4, 10, SCU_PINIO_FAST | SCU_MODE_FUNC4);    // LCD_4:  VSYNC
Chip_SCU_PinMuxSet (4,  8, SCU_PINIO_FAST | SCU_MODE_FUNC4);    // LCD_RS: HSYNC
Chip_SCU_PinMuxSet (7,  5, SCU_MODE_INACT |
                        SCU_MODE_ZIF_DIS | SCU_MODE_FUNC0);    // TEC_C2: EDDC_EN

// Se asigna dirección de los pines de GPIO
Chip_GPIO_SetPinDIROutput (LPC_GPIO_PORT, 2, 8);    // GPIO8:  R2
Chip_GPIO_SetPinDIROutput (LPC_GPIO_PORT, 2, 6);    // LCD3:   R1
Chip_GPIO_SetPinDIROutput (LPC_GPIO_PORT, 2, 5);    // LCD2:   R0
Chip_GPIO_SetPinDIROutput (LPC_GPIO_PORT, 2, 4);    // LCD1:   G2
Chip_GPIO_SetPinDIROutput (LPC_GPIO_PORT, 2, 3);    // T_FIL3: G1
Chip_GPIO_SetPinDIROutput (LPC_GPIO_PORT, 2, 2);    // T_FIL2: G0
Chip_GPIO_SetPinDIROutput (LPC_GPIO_PORT, 2, 1);    // T_FIL1: B1
Chip_GPIO_SetPinDIROutput (LPC_GPIO_PORT, 2, 0);    // T_FIL0: B0
Chip_GPIO_SetPinDIROutput (LPC_GPIO_PORT, 5, 14);   // LCD_4:  VSYNC
Chip_GPIO_SetPinDIROutput (LPC_GPIO_PORT, 5, 12);   // LCD_RS: HSYNC
Chip_GPIO_SetPinDIROutput (LPC_GPIO_PORT, 3, 13);   // TEC_C2: EDDC_EN

// Configuración de I2C
Chip_I2C_Init           (I2C0);
Chip_SCU_I2C0PinConfig  (BOARD_I2C_MODE);
Chip_I2C_SetClockRate   (I2C0, BOARD_I2C_SPEED);
```

Audio

En el listado 4.2 se inicia el puerto I²S para audio PCM de 16 bit, 32 kHz estéreo y se configura un canal DMA para hacer streaming (o descarga continua) del buffer en memoria SRAM hacia el DAC de audio.

Ademas se inicia el pin de salida digital para silenciar el DAC de audio.

LISTADO 4.2: Código de incialización del puerto I²S para audio.

```
#define SOUND_I2S_PORT                  LPC_I2S1
#define SOUND_MIXER_SAMPLE_RATE         32000
#define SOUND_MIXER_BITS                16
#define SOUND_MIXER_CHANNELS            2

// Configuración del puerto I2S.
I2S_AUDIO_FORMAT_T i2sAudioFmt;

i2sAudioFmt.SampleRate      = SOUND_MIXER_SAMPLE_RATE;
i2sAudioFmt.ChannelNumber   = SOUND_MIXER_CHANNELS;
i2sAudioFmt.WordWidth       = SOUND_MIXER_BITS;

Chip_I2S_Init           (SOUND_I2S_PORT);
Chip_I2S_TxConfig       (SOUND_I2S_PORT, &i2sAudioFmt);
Chip_I2S_TxStop         (SOUND_I2S_PORT);
Chip_I2S_DisableMute    (SOUND_I2S_PORT);
Chip_I2S_TxStart        (SOUND_I2S_PORT);

Chip_I2S_DMA_TxCmd      (SOUND_I2S_PORT, I2S_DMA_REQUEST_CHANNEL_1, ENABLE, 4);
Chip_GPDMA_Init         (LPC_GPDMA);

NVIC_DisableIRQ         (DMA_IRQn);
NVIC_SetPriority        (DMA_IRQn, 3);
NVIC_EnableIRQ          (DMA_IRQn);

// ENET_MDC: ~AUDIO_MUTE
Chip_SCU_PinMuxSet          (7, 7, SCU_PINIO_FAST | SCU_MODE_FUNC0);
Chip_GPIO_SetPinDIROutput   (LPC_GPIO_PORT, 3, 15);
```

Puertos de joystick

En el listado 4.3 se inician los pines de entradas y salidas digitales para las señales compartidas de LATCH, CLOCK y las entradas individuales de DATA para los puertos de ambos joysticks.

LISTADO 4.3: Código de incialización de los pines de joysticks.

```
// GPIO1: JOY_LATCH
Chip_SCU_PinMuxSet          (6, 4, SCU_PINIO_FAST | SCU_MODE_FUNC0);
Chip_GPIO_SetPinDIROutput   (LPC_GPIO_PORT, 3, 3);

// GPIO6: JOY_CLK
Chip_SCU_PinMuxSet          (6, 10, SCU_PINIO_FAST | SCU_MODE_FUNC0);
Chip_GPIO_SetPinDIROutput   (LPC_GPIO_PORT, 3, 6);

// GPIO5: JOY1_DATA_IN
Chip_SCU_PinMuxSet          (6, 9, SCU_MODE_INBUFF_EN | SCU_MODE_FUNC4);
Chip_GPIO_SetPinDIRInput    (LPC_GPIO_PORT, 3, 5);

// GPIO7: JOY2_DATA_IN
Chip_SCU_PinMuxSet          (6, 11, SCU_MODE_INBUFF_EN | SCU_MODE_FUNC4);
Chip_GPIO_SetPinDIRInput    (LPC_GPIO_PORT, 3, 7);
```

Módulo Wi-Fi

En el listado 4.4 se muestra un fragmento de la inicialización de pines de la UART3. Esto se realiza como parte de la inicialización del código de la placa «lpcopen_edu_retrociaa_board» en LPCOpen.

En el listado 4.5 se configura la UART3 con los parámetros necesarios para establecer comunicación con el modulo Wi-Fi y se inicia la salida digital para activar dicho modulo.

LISTADO 4.4: Código de incialización de la UART3 (fragmento de «lpcopen_edu_retrociaa_board» en librería LPCOpen).

```
static const PINMUX_GRP_T pinmuxing[] =
{
    // Para mejorar la claridad se omitió copiar aqui el código de inicialización
    // de pines de la EDU-CIAA-NXP

    /* UART3: 232_TX/RX on P1 header */
    {2, 3, (SCU_MODE_INBUFF_EN | SCU_MODE_INACT | SCU_MODE_FUNC2)}, // UART3 TXD
    {2, 4, (SCU_MODE_INBUFF_EN | SCU_MODE_INACT | SCU_MODE_FUNC2)}, // UART3 RXD
};

void Board_SetupMuxing(void)
{
    Chip_SCU_SetPinMuxing(pinmuxing, sizeof(pinmuxing) / sizeof(PINMUX_GRP_T));
}
```

LISTADO 4.5: Código de configuración de la UART3 para el modulo Wi-Fi.

```
#define WIRELESS_UART               LPC_USART3
#define WIRELESS_UART_IRQ           USART3_IRQn
#define WIRELESS_UART_IRQHANDLER    UART3_IRQHandler
#define WIRELESS_UART_BAUD_RATE     115200
#define WIRELESS_UART_DATA_BITS     UART_LCR_WLEN8
#define WIRELESS_UART_PARITY        UART_LCR_PARITY_DIS
#define WIRELESS_UART_STOP_BITS     UART_LCR_SBS_1BIT
#define WIRELESS_UART_CONFIG        (WIRELESS_UART_DATA_BITS \
                                     | WIRELESS_UART_PARITY \
                                     | WIRELESS_UART_STOP_BITS)

Chip_UART_Init          (WIRELESS_UART);
Chip_UART_SetBaudFDR    (WIRELESS_UART, WIRELESS_UART_BAUD_RATE);
Chip_UART_ConfigData    (WIRELESS_UART, WIRELESS_UART_CONFIG);
Chip_UART_TXEnable      (WIRELESS_UART);

// TEC_C1: WM_EN
Chip_SCU_PinMuxSet          (7, 4, SCU_MODE_INACT |
                                 SCU_MODE_ZIF_DIS | SCU_MODE_FUNC0);
Chip_GPIO_SetPinDIROutput   (LPC_GPIO_PORT, 3, 12);
```

4.2.4. Memoria compartida

En el listado 4.6 se observa el linker script que define las tres áreas de memoria compartida entre los núcleos Cortex-M0 y Cortex-M4:

g_framebufferA Un buffer gráfico del esquema de Doublebuffer.

g_framebufferB El otro buffer gráfico en el esquema de Doublebuffer.

g_drvExchange Un área de memoria para intercambiar datos.

Los buffers «g_framebufferA» y «g_framebufferB» son de lectura para ambos núcleos y de escritura exclusiva para el Cortex-M4. El buffer «g_drvExchange» es de lectura y escritura para ambos núcleos.

LISTADO 4.6: Linker script que define las áreas de memoria compartida entre núcleos.

```
__start_RamLoc40_framebuffer    = 0x10080000;
__top_RamLoc40_framebuffer      = __start_RamLoc40_framebuffer + 0x9000;
g_framebufferA                  = __start_RamLoc40_framebuffer;

__start_RamAHB_framebuffer      = 0x20000000;
__top_RamAHB_framebuffer        = __start_RamAHB_framebuffer + 0x9000;
g_framebufferB                  = __start_RamAHB_framebuffer;

__start_RamAHB_exchange         = 0x2000ffe0;
__top_RamAHB_exchange           = __start_RamAHB_exchange + 0x20;
g_drvExchange                   = __start_RamAHB_exchange;
```

4.2.5. Driver de video

En esta subsección se describe la implementación desarrollada para el driver de video y su relación con el firmware de la aplicación.

Memoria de intercambio

Se define un área en SRAM dedicada exclusivamente a compartir datos entre el driver corriendo en el núcleo M0 y la aplicación en el núcleo M4. Esta área de 32 B denominada «g_drvExchange» se estructura con el tipo de dato *struct DRIVER_Exchange* y contiene información como la dirección del frontbuffer, el numero único de cuadro actual, el numero de simulación de scanlines y el estado de los joysticks o gamepads. En el listado 4.7 se observa la definición de la estructura *struct DRIVER_Exchange*.

LISTADO 4.7: Estructura para acceder a los datos en el area de memoria «g_drvExchange».

```
#define DRIVER_FB_WIDTH             256
#define DRIVER_FB_HEIGHT            144
#define DRIVER_FB_SIZE              (DRIVER_FB_WIDTH * DRIVER_FB_HEIGHT)

struct DRIVER_Exchange
{
    volatile uint32_t   vFramebuffer;   // framebuffer base address
    volatile uint32_t   vFrameNumber;   // frames displayed since reset
    volatile uint8_t    vScanlines;     // n = 5-n scanlines, 5-n <= 0 = disabled
    volatile uint8_t    vReserved;      //
    volatile uint8_t    gamepad1;       // gamepad 1 status
    volatile uint8_t    gamepad2;       // gamepad 2 status
};

extern struct DRIVER_Exchange   g_drvExchange;
extern uint8_t                  g_framebufferA [DRIVER_FB_SIZE];
extern uint8_t                  g_framebufferB [DRIVER_FB_SIZE];
```

Sincronización entre núcleos

El punto de sincronización entre el driver de video y la aplicación se implementó en la función *DISPLAY_SwapBuffers()*. Esta función produce que el núcleo M4 espere con un WFI a que el M0 genere la interrupción de Vertical Blanking Interrupt. Al recibirla, el firmware del M4 accede al área compartida en memoria y rápidamente actualiza los valores que necesita pasarle al driver. Aquí es cuando se actualiza la dirección del frontbuffer que es donde el driver busca los pixeles a visualizar en pantalla.

La aplicación tiene algo mas de $16\,\mathrm{ms}$ para llamar a *DISPLAY_SwapBuffers()*. Pasado ese lapso de tiempo el driver pasará a generar un nuevo cuadro de video con el mismo contenido de la antigua dirección del frontbuffer indicada en la memoria compartida.

Sincronización de la señal de video

Para el sincronizado de este componente se decidió utilizar RIT (*Repetitive Interrupt Timer*) [22, p.81] para generar interrupciones periódicas a la misma frecuencia que la señal de sincronismo horizontal o HSYNC: $44\,775{,}28\,\mathrm{Hz}$.

En la interrupción se genera el pulso de sincronismo horizontal y se activa el sincronismo vertical si se esta entre las lineas correspondientes a dicha señal en el rango $(723, 728]$ es decir, las 5 lineas de *Vertical Sync Width*. En este punto se aclara que para hacer mas intuitiva la implementación del driver, se decidió que las lineas de video se cuenten a partir de 1 y no 0 como es habitual en el lenguaje 'C'.

Debido a que la sincronización se genera por un evento de interrupción, el código del driver se estructura en una programación orientada a eventos en donde se asume que HSYNC es el único evento que ocurre y que cualquier instrucción WFI (*Wait For Interrupt*) despierta luego de exactamente transcurrido el periodo del pulso de la señal de HSYNC[2]y con VSYNC mantenido al nivel que corresponda según la linea que se este procesando.

En resumen, WFI es un punto de sincronización con el final del pulso de HSYNC o lo que es lo mismo, el comienzo de una nueva linea de video.

Lineas visibles

Las lineas en el rango $[1, 720]$ contienen pixeles visibles. Al volver del WFI, en estas lineas es necesario generar una espera de tiempo igual al Horizontal Back Porch. Al ser un lapso de tiempo de un par de microsegundos, la espera se implementa con un busy loop con una cantidad de iteraciones calibradas a mano midiendo la señal resultante con osciloscopio.

Inmediatamente después de finalizado el Horizontal Back Porch, se muestran los pixeles del frontbuffer que pertenezcan a la linea que se este procesando tal como se definió en el diseño.

[2]El período $1/44\,775{,}28\,\mathrm{Hz} = 22{,}333\,75\,\mu\mathrm{s}$ es el tiempo que dura una linea completa de pixeles visibles mas el *Horizontal Blanking Interval*, ver tiempo de *Horizontal Total (Line)* en tabla E.2.

Cada pixel del frontbuffer equivale a 5x5 pixeles del modo de video. En cada linea del modo de video se dibujan 5x1 pixeles del frontbuffer. En la ecuación 3.7 se calculó ese tiempo como $TLP5 = 67{,}108\,629\,858\,\text{ns}$.

El dibujo de pixeles se implementa en assembler. Si bien no es completamente determinista, a los fines prácticos se asumirá como cierto que una instrucción de n ciclos de reloj demora $n \cdot MCU_{cycle}$. La constante de duración de un ciclo de reloj fue definida en la ecuación 3.2 como $MCU_{cycle} = 4{,}901\,960\,784\,31\,\text{ns}$.

En la ecuación 4.1 se calcula la cantidad de ciclos de reloj por cada 5 pixeles del modo de video.

$$\begin{aligned} TLP5_{cycles} &= \frac{TLP5}{MCU_{cycle}} \\ &= \frac{67{,}108\,629\,858\,\text{ns}}{4{,}901\,960\,784\,31\,\text{ns}} \\ &= 13{,}690\,160\,491 \end{aligned} \tag{4.1}$$

Como puede observarse, $TLP5_{cycles}$ no es un valor de ciclos de reloj entero y por lo tanto este temporizado de 5 pixeles es imposible de cumplir con el reloj del microcontrolador. Se debe adoptar 13 ciclos, 14 o una combinación de 13 y 14 a lo largo de la linea visible. Debido a que la combinación de 13 y 14 en un esquema repetido de 14-14-13 produjo algunas anomalías perceptibles en los pixeles, en principio se optó por implementar 13 ciclos cada 5 pixeles a fin de no exceder los valores de diseño de Horizontal Porch.

En la ecuación 4.2 se calcula la cantidad real de ciclos de reloj para una linea de pixeles visibles u *Horizontal Active Pixels* correspondiente al valor de diseño.

$$\begin{aligned} \textit{Horizontal Active Pixels Cycles}_{design} &= TLP5_{cycles} \cdot 256 \\ &= 13{,}690\,160\,491 \cdot 256 \\ &= 3504{,}681\,085\,7 \end{aligned} \tag{4.2}$$

En la ecuación 4.3 se calcula la misma cantidad pero con 14 ciclos cada 5 pixeles.

$$\begin{aligned} \textit{Horizontal Active Pixels Cycles}_{14} &= 14 \cdot 256 \\ &= 3584 \end{aligned} \tag{4.3}$$

Finalmente, en la ecuación 4.4 se utiliza el valor de implementación de 13 ciclos cada 5 pixeles.

$$\begin{aligned} \textit{Horizontal Active Pixels Cycles}_{impl} &= 13 \cdot 256 \\ &= 3328 \end{aligned} \tag{4.4}$$

En la tabla 4.3 se listan las diferencias entre el valor de diseño de *Horizontal Active Pixels* definido en la tabla E.2 y el de implementación que utiliza 13 ciclos. Es importante tener en cuenta estos valores ya que adoptar 13 ciclos de reloj por cada 5 pixeles dibujados acorta el largo total de la linea visible haciéndola sensiblemente menor a la de diseño.

TABLA 4.3: Diferencia entre el valor de diseño para *Horizontal Active Pixels* y el utilizado en la implementación

Horizontal Active Pixels	Valor diseño	Valor implementación (13 ciclos)	Diferencia
Ciclos de reloj	3504,68	3328	−176,68
Tiempo	17,179 80 µs	16,313 72 µs	−0,866 08 µs
Cobertura de señal	100 %	94,95 %	−5,05 %

En la tabla 4.4 se listan las mismas diferencias pero contra el calculo de 14 ciclos de reloj.

TABLA 4.4: Diferencia entre el valor de diseño para *Horizontal Active Pixels* y el calculado con 14 ciclos de reloj.

Horizontal Active Pixels	Valor diseño	Valor para 14 ciclos de reloj	Diferencia
Ciclos de reloj	3504,68	3584	79,32
Tiempo	17,179 80 µs	17,568 62 µs	0,388 82 µs
Cobertura de señal	100 %	102,26 %	2,26 %

Comparando las tablas 4.3 y 4.4 y observando el resultado de los ensayos, queda claro que 14 ciclos es una mejor elección a implementar en una futura versión del driver.

La implementación del dibujo de una linea de video puede verse en el listado 4.8.

LISTADO 4.8: Código fuente del driver de video

```
1  #define GPIO2_MASK          0xFFFFFE80
2  #define GPIO2_MASK_ADDR     0x400F6088
3  #define GPIO2_MPIN_ADDR     0x400F6188
4
5  .syntax unified
6  .thumb
7  .text
8
9  @ r0 = direccion de memoria del framebuffer
10 .global DisplayLine
11 .thumb_func
12     DisplayLine:
13           push {r4-r5}
14           @ Guarda mascara original en r4 y setea nueva.
15           ldr  r1, =GPIO2_MASK_ADDR
16           ldr  r4, [r1]
17           ldr  r2, =GPIO2_MASK
18           str  r2, [r1]
19           @ Direccion de MPIN en r2.
20           ldr  r2, =GPIO2_MPIN_ADDR
21           @ Procesa una linea de 256 pixels logicos, 1 pixel por iteracion.
22           ldr  r5, =256
23       .displayLinePixel:
24           @ Se muestran 5 pixels de 1280 por cada pixel logico de 5x1 en el
25           @ framebuffer. La duracion total de la linea visible de 1280 pixels es
26           @ 17,17980 us. Como cada grupo de 5 pixels no da un numero redondo en
27           @ ciclos de reloj del M0 (13,69) se decidio utilizar 13 ciclos por
28           @ grupo de 5x1 pixeles. Esto produce que la linea visible dure menos:
29           @ 16,31372 us en total, pero en principio genera resultados mas
```

```
30          @ consistentes que compensar ciclos de reloj intercalando 13 y 14
31          @ ciclos para cumplir con la duracion de linea.
32          nop                             @ +1
33          ldrb r1, [r0, #0]               @ +2          Carga pixel
34          lsrs r3, r1, #7                 @ +1          Rxxxxxxx  ->  0000000R
35          lsls r3, #8                     @ +1          0000000R00000000
36          orrs r1, r3                     @ +1          0000000RxRRGGGBB
37          str  r1, [r2]                   @ +2          Setea pixel en GPIO2
38          adds r0, #1                     @ +1          Incrementa buffer
39          subs r5, #1                     @ +1          Una iteracion menos
40          bne  .displayLinePixel          @ +3 (13)     Hsta no procesar todo
41      .endDisplayLine:                    @ +1 (bne tomado)
42          @ Setea salidas a 0. Los pixels dentro del blanking deben ser negros.
43          ldr  r1, =0                     @ +2
44          str  r1, [r2]                   @ +2
45          @ Devuelve la mascara original
46          ldr  r1, =GPIO2_MASK_ADDR       @ +2
47          str  r4, [r1]                   @ +2
48          pop  {r4-r5}                    @ +4
49          bx   lr                         @ +1
```

Lineas no visibles

Las lineas no visibles no necesitan generar la espera del Horizontal Back Porch.

En la linea 721 se utiliza el tiempo de esa linea para actualizar el estado de los joysticks. La interrupción de Vertical Blanking Interval (VBI) se genera en la linea 722. Esa linea es el punto de sincronización entre el driver y la aplicación y en este punto la aplicación ya debería haber llamado a la función *DISPLAY_SwapBuffers()*.

La función *DISPLAY_SwapBuffers()* tiene desde la linea 722 hasta la 748 o un tiempo de $22{,}333\,75\,\mu s * 26 = 580{,}67\,\mu s$ de acceso exclusivo a la memoria compartida para manipularla con total seguridad.

Al final de la linea 748 (la ultima linea del cuadro de video actual) el driver accede a la memoria compartida para incrementar el numero de cuadro y actualizar sus variables internas con la dirección del frontbuffer y simulación de scanlines. Estos datos se usan para la próxima linea 1 del nuevo cuadro de video.

Código fuente

En el listado 4.9 se observa el código fuente del driver de video. Para implementar un modo de video distinto solo se requiere modificar los valores de las constantes con prefijo «DRIVER_».

LISTADO 4.9: Código fuente del driver de video

```
1  static uint32_t g_vFramebuffer;
2  static uint32_t g_vFBOffset     = 0;
3  static uint32_t g_vLineNum      = 1;
4  static uint32_t g_vLineDup      = 0;
5  static uint32_t g_vScanlines;
6  static uint32_t g_vFrameCount   = 0;
7  static uint8_t  g_vGamepad1     = 0xFF;
8  static uint8_t  g_vGamepad2     = 0xFF;
9
10
11 static void initHardware ()
12 {
```

```
13      SystemCoreClockUpdate ();
14
15      DRIVER_SET_SYNC_SIGNAL (HSYNC, OFF);
16      DRIVER_SET_SYNC_SIGNAL (VSYNC, OFF);
17
18      DRIVER_SET_JOY_SIGNAL (JOY_CLOCK, LOW);
19      DRIVER_SET_JOY_SIGNAL (JOY_LATCH, LOW);
20
21      g_vFramebuffer  = g_drvExchange.vFramebuffer;
22      g_vScanlines    = g_drvExchange.vScanlines;
23
24      g_drvExchange.gamepad1  = g_vGamepad1;
25      g_drvExchange.gamepad2  = g_vGamepad2;
26
27      // Activate Repetitive Interrupt Timer (RIT) for periodic IRQs
28      Chip_RIT_Init       (LPC_RITIMER);
29
30      // CLK_MX_RITIMER Clock Rate = 204000000
31      const uint32_t Cmp_value = Chip_Clock_GetRate(CLK_MX_RITIMER) /
32                                                              DRIVER_HSYNC_HZ;
33      Chip_RIT_SetCOMPVAL (LPC_RITIMER, Cmp_value);
34      Chip_RIT_EnableCTRL (LPC_RITIMER, RIT_CTRL_ENCLR);
35      Chip_RIT_Enable     (LPC_RITIMER);
36
37      // Enable IRQ for RIT
38      NVIC_SetPriority    (RITIMER_IRQn, 0);
39      NVIC_EnableIRQ      (RITIMER_IRQn);
40  }
41
42
43  void RIT_IRQHandler ()
44  {
45      if (!(LPC_RITIMER->CTRL & RIT_CTRL_INT))
46      {
47          return;
48      }
49
50      Chip_RIT_ClearInt       (LPC_RITIMER);
51      NVIC_ClearPendingIRQ    (RITIMER_IRQn);
52
53      // HSYNC
54      DRIVER_SET_SYNC_SIGNAL (HSYNC, ON);
55      BusyLoop (DRIVER_HSYNC_WIDTH);
56      DRIVER_SET_SYNC_SIGNAL (HSYNC, OFF);
57
58      // VSYNC
59      if (g_vLineNum > DRIVER_VSYNCSTART && g_vLineNum <= DRIVER_VSYNCEND)
60      {
61          DRIVER_SET_SYNC_SIGNAL (VSYNC, ON);
62      }
63      else
64      {
65          DRIVER_SET_SYNC_SIGNAL (VSYNC, OFF);
66      }
67  }
68
69
70  int main ()
71  {
72      initHardware ();
73
74      while (1)
75      {
76          // Visible Pixels
77          if (g_vLineNum <= DRIVER_VDISP)
78          {
79              if (g_vLineDup < g_vScanlines)
80              {
81                  __WFI ();
82                  BusyLoop (DRIVER_BP_WIDTH);
83                  // Duplica linea en framebuffer
84                  DisplayLine (g_vFBOffset);
```

```
85              }
86              else
87              {
88                  __WFI ();
89                  BusyLoop (DRIVER_BP_WIDTH);
90                  // Efecto de "scanline"
91              }
92          }
93          else
94          // En Vertical Blanking Interval
95          {
96              // Primer linea de VBI
97              if (g_vLineNum == DRIVER_VDISP + 1)
98              {
99                  __WFI ();
100                 DRIVER_SET_JOY_SIGNAL (JOY_LATCH, HIGH);
101                 BusyLoop (DRIVER_JOY_LATCH_WIDTH);
102                 DRIVER_SET_JOY_SIGNAL (JOY_LATCH, LOW);
103                 BusyLoop (DRIVER_JOY_LATCH_WIDTH);
104
105                 g_vGamepad1 = DRIVER_GET_JOY_SIGNAL (JOY1);
106                 g_vGamepad2 = DRIVER_GET_JOY_SIGNAL (JOY2);
107
108                 for (uint32_t i = 0; i < 7; ++i)
109                 {
110                     DRIVER_SET_JOY_SIGNAL (JOY_CLOCK, HIGH);
111                     BusyLoop (DRIVER_JOY_CLOCK_WIDTH);
112                     DRIVER_SET_JOY_SIGNAL (JOY_CLOCK, LOW);
113                     BusyLoop (DRIVER_JOY_CLOCK_WIDTH);
114
115                     g_vGamepad1 <<= 1;
116                     g_vGamepad1 |= DRIVER_GET_JOY_SIGNAL (JOY1);
117                     g_vGamepad2 <<= 1;
118                     g_vGamepad2 |= DRIVER_GET_JOY_SIGNAL (JOY2);
119                 }
120
121                 g_drvExchange.gamepad1 = g_vGamepad1;
122                 g_drvExchange.gamepad2 = g_vGamepad2;
123             }
124             else
125             // Mayor o igual a la segunda linea de VBI
126             {
127                 __WFI ();
128                 // Si se procesaron todas las lineas visibles y la primer linea
129                 // del vertical blanking, termina toda operacion pendiente de
130                 // acceso a memoria y genera una interrupcion al core M4.
131                 // (la 1er linea dentro del vertical blanking se utiliza para
132                 // obtener el estado de los joysticks/gamepads).
133                 if (g_vLineNum == DRIVER_VDISP + 2)
134                 {
135                     __DSB ();
136                     __SEV ();
137                 }
138             }
139         }
140
141         // Avanza a la proxima linea
142         if (++ g_vLineNum > DRIVER_VTOTAL)
143         {
144             // Si se procesaron todas inicia un nuevo frame tomando los datos de
145             // exchange
146             g_drvExchange.vFrameNumber = ++ g_vFrameCount;
147
148             g_vFramebuffer  = g_drvExchange.vFramebuffer;
149             g_vScanlines    = g_drvExchange.vScanlines;
150             g_vLineNum      = 1;
151             g_vFBOffset     = g_vFramebuffer;
152             g_vLineDup      = 0;
153         }
154         // Se fija si debe procesar la proxima linea del framebuffer o se sigue
155         // duplicando la actual
156         else if (++ g_vLineDup > 4)
```

```
157         {
158             g_vLineDup = 0;
159             g_vFBOffset += 256;
160         }
161     }
162
163     return 0;
164 }
```

4.2.6. Funciones gráficas

Las funciones gráficas se encargan de dibujar diferentes elementos en el frontbuffer en una posición determinada. La función gráfica mas simple es poner o leer un pixel y las mas complejas implican dibujar texto o gráficos con mascaras y clipping, conceptos que se explicaran a continuación.

Sistema de coordenadas

El sistema de coordenadas de la pantalla puede observarse en la figura 4.7. El origen se sitúa en la esquina superior izquierda de la misma. Este origen se debe a la dirección de escaneo del driver de video (sección 3.2.12) y del monitor (*raster scan*, sección 3.2.10).

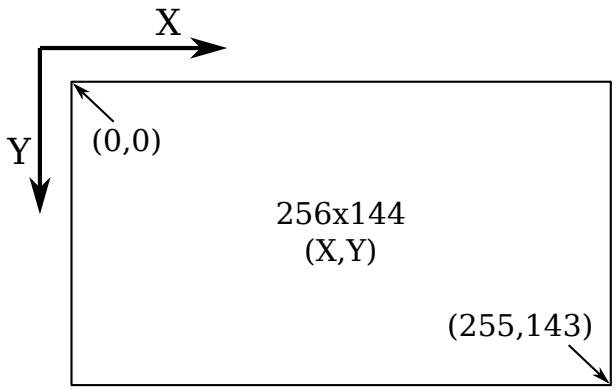


FIGURA 4.7: Sistema de coordenadas de la pantalla.

Las funciones gráficas acceden a un pixel del backbuffer mediante las coordenadas X e Y. En la tabla 4.5 se lista las características de dichas coordenadas.

TABLA 4.5: Coordenadas X e Y para acceder a un pixel del backbuffer

Coordenada	Rango	Descripción
X	$[0, 255]$	Desplazamiento en la linea seleccionada
Y	$[0, 143]$	Linea seleccionada

En la ecuación 4.5 se observa la forma de direccionar un pixel en el área de memoria lineal del backbuffer utilizando coordenadas bidimensionales.

$$BUFFER_{Offset}(X, Y) = Y * 256 + X \tag{4.5}$$

Mascaras

La mascara es un mapa de bits o *bitmap* del mismo tamaño que el elemento a dibujar y que sirve para indicar áreas invisibles en ese elemento. Un bit en 1 indica que el pixel en esa posición relativa[3]debe dibujarse y un bit en 0 indica que no.

En la figura 4.8 se observa un pixmap, su mascara y el dibujo resultante sobre el framebuffer.

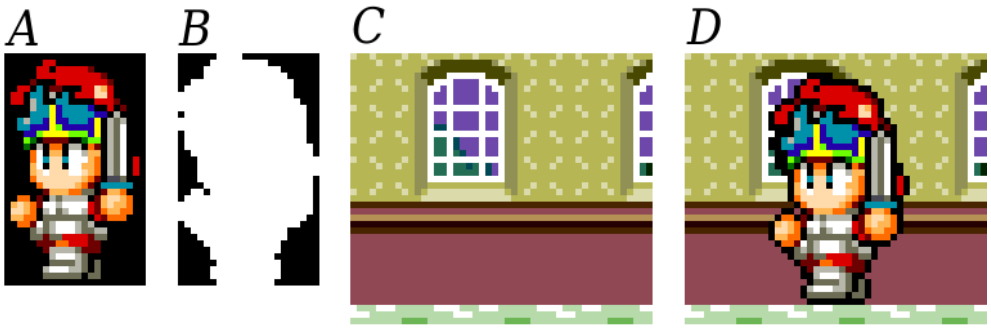


FIGURA 4.8: Sistema de pixeles invisibles por mascara. (A) Pixmap original. (B) Mascara binaria o bitmap. (C) Contenido en backbuffer. (D) Resultado al dibujar el pixmap con su mascara. Edición de imagen del videojuego *Wonder Boy in Monster World* (1991) para consola SEGA Mega Drive.

Clipping

La operación consiste en recortar un elemento a dibujar según los limites definidos en un área de clipping de modo que el dibujo no modifique los pixeles del backbuffer que queden fuera de esa área.

El área puede ser igual a toda la pantalla con el objetivo de recortar partes de un elemento que cruza los bordes quedando parcial o totalmente fuera de la misma. También puede achicarse a un área de menor tamaño en cualquier posición dentro de la pantalla.

Dibujar un elemento que cruce los limites de la pantalla sin utilizar clipping genera errores. Por ejemplo, dibujar cruzando el margen inferior puede generar un *HardFault*[4]por acceder a un offset de memoria que supera el tamaño asignado. Concretamente si el buffer tiene un tamaño de $256x144 = 36\,864\,\mathrm{B}$, en lenguaje 'C' un offset o indice a ese buffer es valido solo desde 0 hasta 36863 inclusive. Acceder una linea mas allá de la ultima dentro de la pantalla implica un offset de $Y \cdot 256 + X = 144 \cdot 256 + 0 = 36864$ o un buffer overflow de $36864 - 36863 = 1\,\mathrm{B}$.

En la figura 4.9 puede apreciarse el efecto de dibujar un pixmap ubicado parcialmente fuera del área de clipping.

[3]Cada bit de la mascara es un byte en el pixmap o frontbuffer.

[4]Un estado de falla del microcontrolador. El manejador correcto sería *MemManage*, pero si no esta definido en el firmware cae en HardFault por defecto.

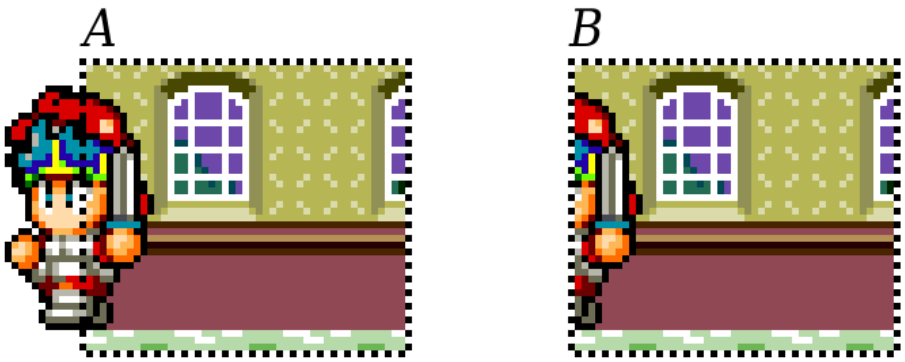


FIGURA 4.9: Efecto del clipping al dibujar en el framebuffer. (A) Area de clipping y ubicación de un pixmap parcialmente fuera del área, previo al dibujado. (B) Resultado del dibujado. Edición de imagen del videojuego *Wonder Boy in Monster World* (1991) para consola SEGA Mega Drive.

Listado de funciones implementadas

A continuación se listan algunas funciones implementadas durante el desarrollo del firmware. La extensión de esta memoria y el tiempo planificado para su concreción no permite comentar los fundamentos, las técnicas y la implementación al detalle. Pero se brinda una breve reseña y código fuente comentado.

- Capacidad de definir un área de clipping de cualquier tamaño y posición en la pantalla.
- Dibujo de texto en codificación UTF-8 de longitud y cantidad de lineas variable, caracteres de 8x8 pixeles y color a elección. El tipo de letra por defecto incluye los siguientes bloques de caracteres del plano Unicode: Basic Latin (U+0020 - U+007F), Latin 1 Supplement (U+00A0 - U+00FF), Greek-Coptic (U+0390 - U+03C9), Box Drawing (U+2500 - U+257F), Block Elements (U+2580 - U+259F), Hiragana (U+3040 - U+309F).
- Dibujo de tiles de 8x8 pixeles y soporte para tilemaps. Un tile es esencialmente un pixmap de 8x8 con mascara de transparencia opcional. La diferencia entre tiles y pixmaps es que en el primer caso las imágenes se generan combinando diferentes tiles de 8x8 pixeles. La información de cuales tiles constituyen una imagen y en que ubicación se disponen se guarda en un *tilemap*. La extensión o tamaño de un tilemap esta solo limitado por el máximo valor entero de un numero de 32 bit y la memoria de almacenamiento disponible. Las funciones de dibujo extraen solo la información necesaria y dibujan los tiles visibles según la posición del tilemap en pantalla. Con esta técnica se pueden crear mapas de videojuegos o pixmaps enormes de manera eficiente.
- Almacenamiento en soportes MicroSD y USB Mass Storage con librería FatFs preparada para múltiples unidades de disco (SDC: y USB:). En USB se soporta enumeración e inicialización al vuelo. Con este código se contribuyó al proyecto de firmware cese-edu-ciaa-template [21], futuro firmware-v3 del Proyecto CIAA (Computadora Industrial Abierta Argentina).

- Dibujo de lineas con un algoritmo original desarrollado por el autor de esta memoria, mas eficiente que el clásico algoritmo de Bresenham. En el listado 4.10 se muestra una versión del mismo sin comprobación de parámetros y utilizando un solo color básico.
- Las funciones que toman un color soportan un tipo de dato especial que también permite especificar paletas de colores cíclicas o un color especifico dentro de una paleta.
- Lista cíclica que acepta tamaño solo en base 2 pero implementada con un algoritmo simple y eficiente, también creación original del autor de esta memoria. En el listado 4.11 puede verse la implementación de la lista cíclica.

Las funciones de tiles y texto soportan un área de clipping arbitraria. Las lineas aun no. Se planifica implementar el algoritmo Liang-Barsky para descubrir los puntos de intersección entre la linea y el área de clipping.

LISTADO 4.10: Dibujo de lineas implementado con un algoritmo de alta performance creado por el autor de esta memoria.

```
1  inline static void swap_i16 (int16_t *a, int16_t *b)
2  {
3      const int16_t A = *a;
4      *a = *b;
5      *b = A;
6  }
7
8  void LINE_Draw (int16_t x1, int16_t y1, int16_t x2, int16_t y2, uint8_t color)
9  {
10     // Valor absoluto de la longitud del recorrido horizontal y vertical
11     const uint32_t H = (x2 > x1)? x2 - x1 : x1 - x2;
12     const uint32_t V = (y2 > y1)? y2 - y1 : y1 - y2;
13
14     // el formato de m es fixed point 8.24
15     uint32_t    m;
16     uint32_t    steps;
17     int32_t     step;
18     int32_t     delta;
19
20     // Recorre en un paso siempre positivo (step) la longitud mayor; de a un
21     // pixel por iteracion en H o una linea por iteracion en V. (delta) es el
22     // salto de linea en H o salto de pixel en V cuando sucesivas acumulaciones
23     // de (m) en (ds) llegan a representar un valor entero. Esta magnitud es
24     // positiva o negativa segun la orientacion de la linea.
25     if (H >= V)
26     {
27         if (x1 > x2)
28         {
29             swap_i16 (&x1, &x2);
30             swap_i16 (&y1, &y2);
31         }
32
33         m       = (V << 24) / H;
34         steps   = H;
35         step    = 1;
36         delta   = (y1 > y2)? -DRIVER_FB_WIDTH : DRIVER_FB_WIDTH;
37     }
38     else
39     {
40         if (y1 > y2)
41         {
42             swap_i16 (&x1, &x2);
43             swap_i16 (&y1, &y2);
44         }
45
46         m       = (H << 24) / V;
47         steps   = V;
```

```
48          step    = DRIVER_FB_WIDTH;
49          delta   = (x1 > x2)? -1 : 1;
50      }
51
52      // (fb) es la posicion inicial en el backbuffer
53      uint8_t *fb = &DISPLAY_Get()->backBuffer[x1 + y1 * DRIVER_FB_WIDTH];
54      uint32_t ds = 0x007FFFFF;   // 0.5
55      const int32_t Zd[2] = { 0, delta };
56
57      // los pixeles se van colocando en posicion (fb) modificada progresivamente
58      // por (step) y, si (ds) llega a entero, tambien por (delta).
59      do
60      {
61          *fb = color;
62          ds += m;
63          fb += step;
64          fb += Zd[ds >> 24]; // ds es entero? suma delta
65          ds &= 0x00FFFFFF;   // siempre elimina valor entero
66      }
67      while (steps --);
68
69      return true;
70  }
```

LISTADO 4.11: Lista ciclica implementada con un algoritmo simple y eficiente creado por el autor de esta memoria.

```
1   // cyclic.h ////////////////////////////////////////////////////////////////
2   #pragma once
3   #include <stdint.h>
4   #include <stdbool.h>
5
6   struct CYCLIC
7   {
8       uint8_t     *data;
9       uint32_t    capacity;
10      // in: producer / data input
11      uint32_t    inIndex;
12      // out: consumer / data output
13      uint32_t    outIndex;
14      // No parte del algoritmo, solo para recolectar estadisticas.
15      uint32_t    reads;
16      uint32_t    writes;
17      uint32_t    overflows;
18      uint32_t    peeks;
19      uint32_t    discards;
20  };
21  // Prototipos de funciones omitidos aqui para ahorrar espacio
22
23
24  // cyclic.c ////////////////////////////////////////////////////////////////
25  #include "cyclic.h"
26  #include <string.h>
27
28  bool CYCLIC_Init (struct CYCLIC *c, uint8_t *data, uint32_t capacity)
29  {
30      if (!c || !data || !capacity)
31      {
32          return false;
33      }
34
35      // Capacity debe ser potencia de 2
36      if (capacity & (capacity - 1))
37      {
38          return false;
39      }
40
41      memset (c, 0, sizeof(struct CYCLIC));
42
43      c->data       = data;
44      c->capacity   = capacity;
```

```
45      return true;
46  }
47
48
49  uint32_t CYCLIC_Pending (struct CYCLIC *c)
50  {
51      if (!c)
52      {
53          return 0;
54      }
55
56      return (c->inIndex - c->outIndex) & (c->capacity - 1);
57  }
58
59
60  bool CYCLIC_In (struct CYCLIC *c, const uint8_t data)
61  {
62      return CYCLIC_InFromBuffer (c, &data, 1);
63  }
64
65
66  bool CYCLIC_InFromBuffer (struct CYCLIC *c, const uint8_t *data, uint32_t size)
67  {
68      if (!c || !data || !size)
69      {
70          return false;
71      }
72
73      const uint32_t Pending = CYCLIC_Pending (c);
74
75      for (uint32_t i = size; i; --i)
76      {
77          c->data[c->inIndex] = *data ++;
78          c->inIndex = (c->inIndex + 1) & (c->capacity - 1);
79      }
80
81      c->writes += size;
82      if (size > c->capacity - Pending)
83      {
84          c->overflows += size - (c->capacity - Pending);
85      }
86
87      return true;
88  }
89
90
91  bool CYCLIC_Out (struct CYCLIC *c, uint8_t *data)
92  {
93      return CYCLIC_OutToBuffer (c, data, 1);
94  }
95
96
97  // Antes llamar a CYCLIC_Pending() para conocer el maximo de datos que puede
98  // devolverse.
99  bool CYCLIC_OutToBuffer (struct CYCLIC *c, uint8_t *data, uint32_t count)
100 {
101     if (!c || !data || !count)
102     {
103         return false;
104     }
105
106     const uint32_t Pending = CYCLIC_Pending (c);
107     if (!Pending)
108     {
109         return true;
110     }
111
112     if (count > Pending)
113     {
114         count = Pending;
115     }
116
```

```
117     c->reads += count;
118
119     while (count --)
120     {
121         *data ++ = c->data[c->outIndex];
122         c->outIndex = (c->outIndex + 1) & (c->capacity - 1);
123     }
124
125     return true;
126 }
127
128
129 // Antes llamar a CYCLIC_Pending() para conocer la cantidad de datos disponibles
130 uint8_t CYCLIC_Peek (struct CYCLIC *c, uint32_t offset)
131 {
132     if (!c)
133     {
134         return 0;
135     }
136
137     ++ c->peeks;
138     return c->data[(c->outIndex + offset) & (c->capacity - 1)];
139 }
140
141
142 bool CYCLIC_DiscardPending (struct CYCLIC *c)
143 {
144     if (!c)
145     {
146         return false;
147     }
148
149     const uint32_t Pending = CYCLIC_Pending (c);
150     c->outIndex = (c->outIndex + Pending) & (c->capacity - 1);
151     c->discards += Pending;
152     return true;
153 }
```

Ejemplos

En el listado 4.12 se aporta código de ejemplo para una mejor comprensión de la forma de leer y escribir pixeles en el backbuffer. Al ser un solo pixel, la operación de corroborar que las coordenadas no estén fuera de rango equivale a aplicar un clipping en pantalla completa.

LISTADO 4.12: Código de ejemplo de funciones putpixel y getpixel

```
1  bool checkRange (int32_t x, int32_t y)
2  {
3      // Comprueba que X este dentro del rango permitido.
4      if (x < 0 || x > 255)
5      {
6          return false;
7      }
8
9      // Comprueba que Y este dentro del rango permitido.
10     if (y < 0 || y > 143)
11     {
12         return false;
13     }
14
15     return true;
16 }
17
18
19 uint32_t getOffset (int32_t x, int32_t y)
20 {
```

```
21      // Devuelve offset a memoria lineal segun ecuación 4.5.
22      return y * 256 + x;
23  }
24
25
26  uint8_t getPixel (int32_t x, int32_t y)
27  {
28      // Comprueba que las coordenadas esten en el rango permitido.
29      if (!CheckRange(x, y))
30      {
31          return 0;
32      }
33
34      // Devuelve pixel en coordenadas (X,Y).
35      return DISPLAY_Get()->backBuffer[getOffset(x, y)];
36  }
37
38
39  void setPixel (int32_t x, int32_t y, uint8_t color)
40  {
41      // Comprueba que las coordenadas esten en el rango permitido.
42      if (!CheckRange(x, y))
43      {
44          return;
45      }
46
47      // Asigna pixel en coordenadas (X,Y).
48      DISPLAY_Get()->backBuffer[getOffset(x, y)] = color;
49  }
```

Capítulo 5

Ensayos y Resultados

5.1. Pruebas funcionales de hardware

El objetivo de estas pruebas fue corroborar el correcto funcionamiento del conjunto EDU-CIAA-NXP/RETRO-CIAA y validar la implementación del diseño de hardware. Se corroboró que las tensiones y temporizados de las señales eléctricas estuviesen dentro de los parámetros de diseño.

En las mediciones con osciloscopio se indica la señal en mayúscula y entre paréntesis el canal utilizado para su medición, por ejemplo CLOCK(1). En las capturas de pantalla, el canal de cada señal se indica en el borde izquierdo de la misma.

5.1.1. Lecturas de tensión

Se realizan lecturas de tensión de alimentación mediante multimetro para corroborar el funcionamiento normal del conjunto EDU-CIAA-NXP/RETRO-CIAA. Las lecturas se tomaron utilizando los pines de expansión de la EDU-CIAA-NXP conectados en la RETRO-CIAA a través del puerto «P4». Los pines utilizados fueron pin 1 para 3,3 V, pin 2 para 5 V y pin 39 para GND.

Para la tensión de 3,3 V se tomaron 1897 mediciones automáticas realizadas en modo histograma a razón de 3 mediciones por segundo. Esta tensión no presentó anomalías durante el tiempo del ensayo. El resultado puede verse en la figura 5.1.

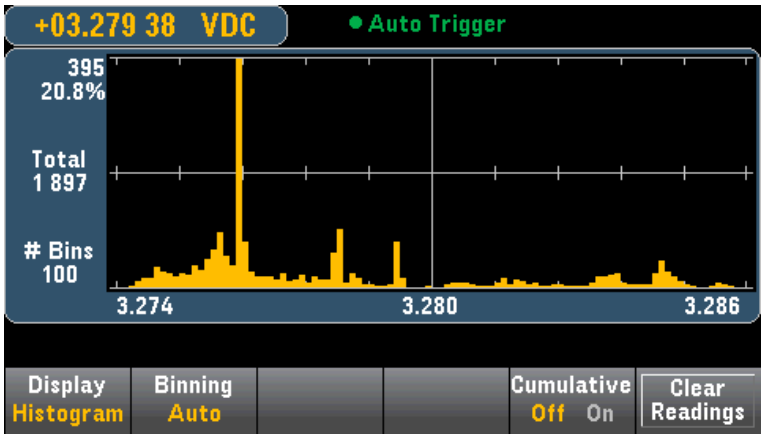


FIGURA 5.1: Lecturas de la tensión de alimentación de 3,3 V.

En la figura 5.2 puede observarse el mismo ensayo para la tensión de 5 V pero tomando 1906 mediciones automáticas. El nivel respondería a la caída de tensión

en el diodo Schottky PMEG3020EH colocado a la entrada de la alimentación de 5 V de la EDU-CIAA-NXP.

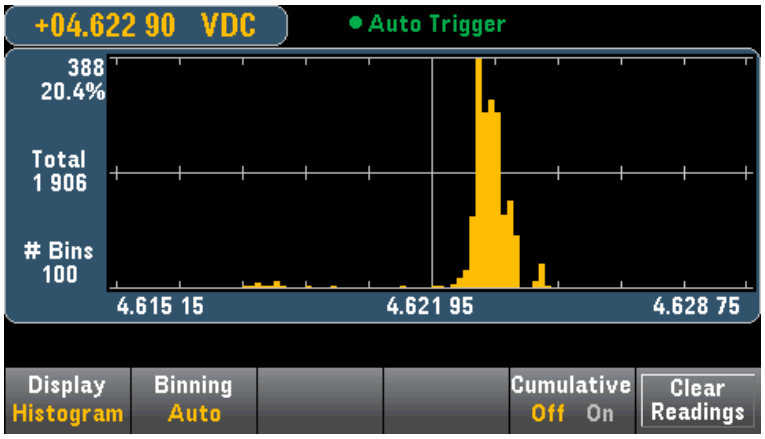


FIGURA 5.2: Lecturas de la tensión de alimentación de 5 V.

5.1.2. Consumo de energía

Se utiliza una fuente de laboratorio para alimentar al conjunto EDU-CIAA-NXP/RETRO-CIAA y corroborar su funcionamiento normal y consumo con y sin el modulo Wi-Fi activado.

La fuente utilizada mantiene una tensión constante de 5 V e informa de las variaciones de consumo de corriente.

En la figura 5.3 se observa el consumo de energía con el modulo Wi-Fi apagado: 266 mA en 5 V o 1,33 W.

FIGURA 5.3: Consumo de corriente con alimentación de 5 V y modulo Wi-Fi apagado.

El consumo de energía con el modulo Wi-Fi encendido es de 296 mA en 5 V o 1,48 W. Esto puede verse en la figura 5.4.

En ambos casos el valor capturado es una media. Ambos valores fluctúan constantemente en ±15 mA.

FIGURA 5.4: Consumo de corriente con alimentación de 5 V y modulo Wi-Fi encendido.

5.1.3. Señales de video

El objetivo de estos ensayos es corroborar que las señales de video se encuentran dentro de las especificaciones de diseño.

Sincronismo

Se realizan mediciones con osciloscopio de las señales de sincronismo. Todos los parámetros medidos se comprueban sobre valores definidos la etapa de diseño presentes en la subsección 3.2.10.

En la tabla 5.1 se observan los parámetros medidos con osciloscopio, el numero de figura de la captura de pantalla de la medición, la transcripción de la medición y el valor de diseño para su comparación.

TABLA 5.1: Parámetros de sincronismo medidos con osciloscopio y comparación con valores de diseño.

Parámetro	Figura	Valor medido	Valor de diseño
Vertical Front Porch	5.5	67 µs	67 µs
Vertical Front Porch (lineas)	5.5	3 lineas	3 lineas
Vertical Sync Width	5.6	111 µs	111 µs
VSYNC	5.6	59,848 Hz	59,86 Hz
VSYNC (lineas)	5.6	5 lineas	5 lineas
Vertical Back Porch	5.7	0,447 ms	0,446 67 ms
Vertical Back Porch (lineas)	5.7	20 lineas	20 lineas
Frame period	5.8	16,6 ms	16,7 ms
Horiz. Total - Horiz. Sync Width	5.9	20,6 µs	20,615 77 µs
Horiz. Front Porch	5.10	1,92 µs	0,858 99 µs
$\overline{\text{HSYNC}}$	5.10	44,7662 kHz	44,77 kHz
Horiz. Sync Width	5.11	1,71 µs	1,717 98 µs
Horiz. Back Porch	5.12	2,43 µs	2,576 97 µs

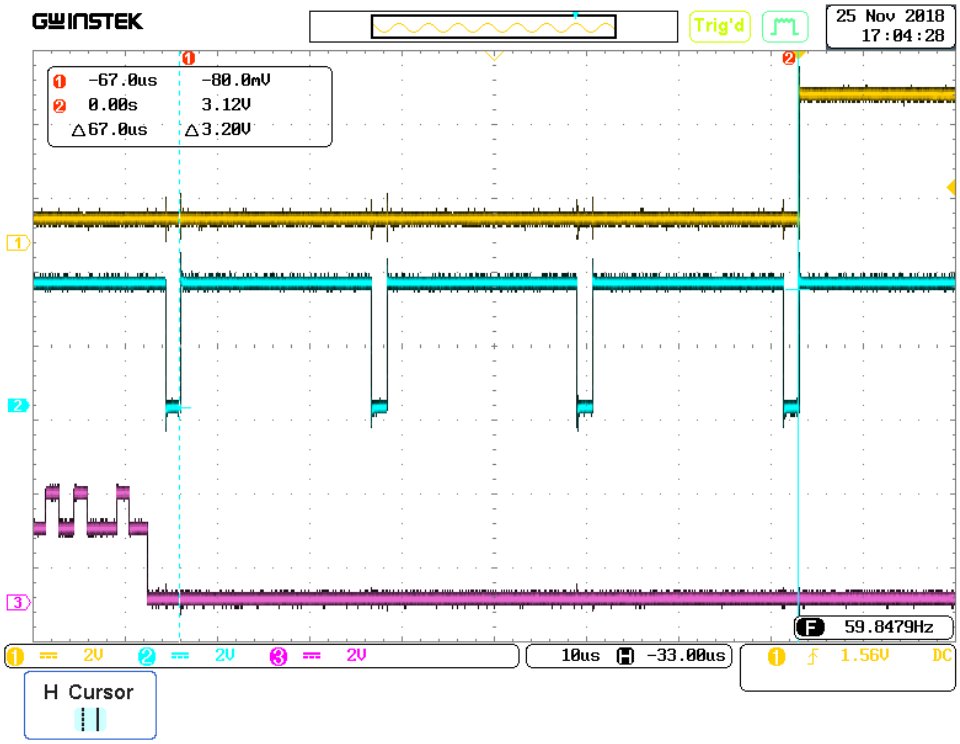


FIGURA 5.5: Duración del Vertical Front Porch de la señal VSYNC(1). Se muestran señales $\overline{\text{HSYNC}}$(2) y componente R(3).

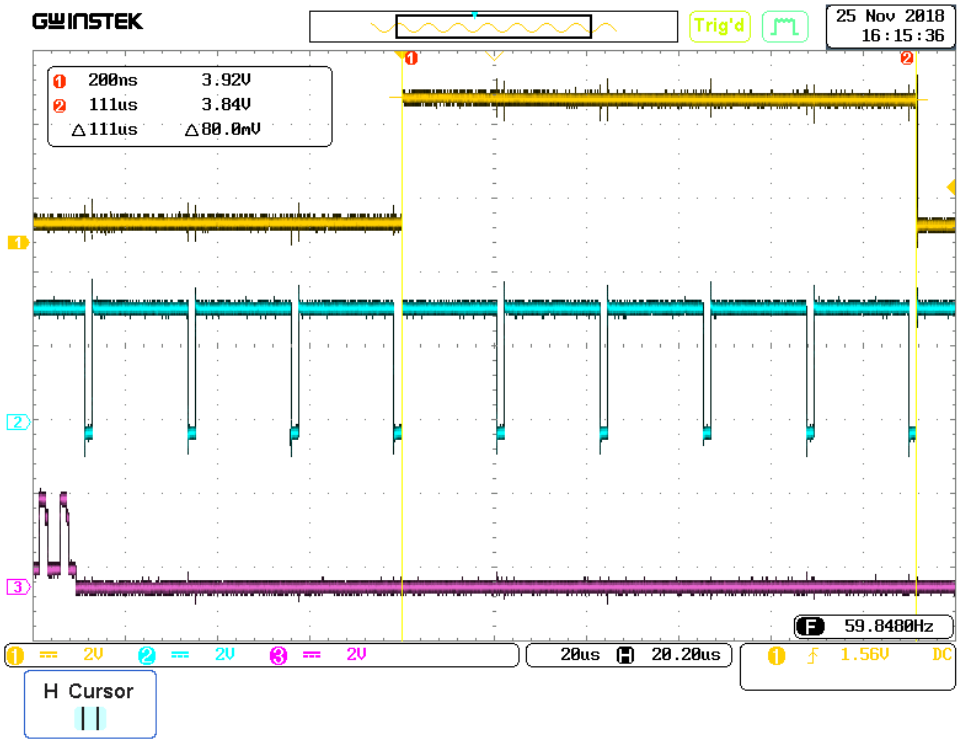


FIGURA 5.6: Ancho de pulso de la señal VSYNC(1). Se muestran señales $\overline{\text{HSYNC}}$(2) y componente R(3).

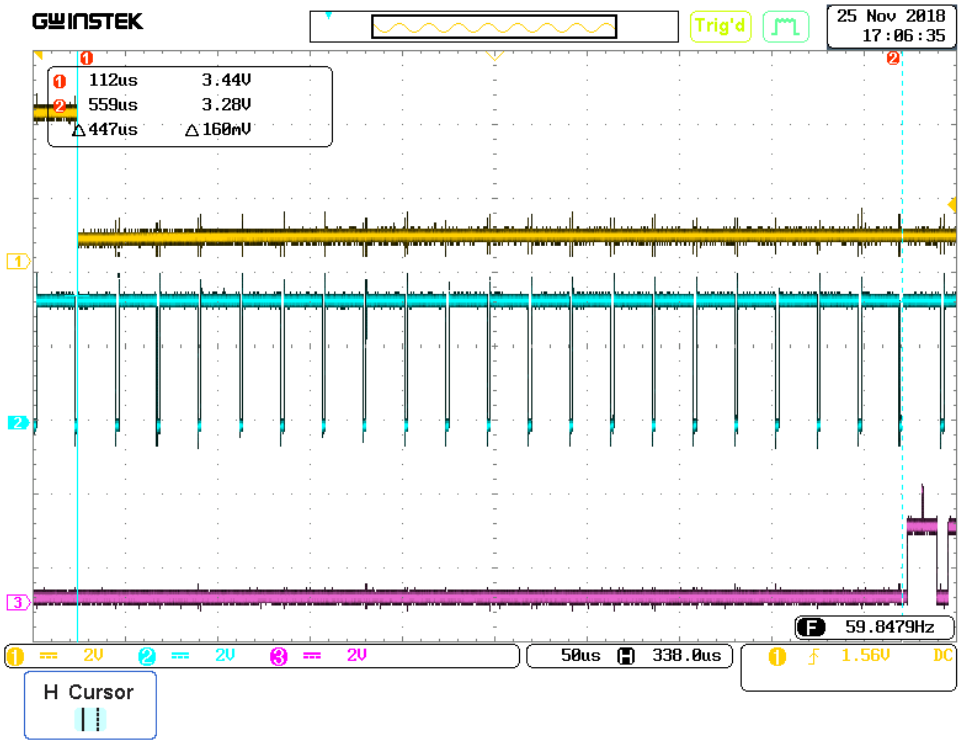


FIGURA 5.7: Duración del Vertical Back Porch de la señal VSYNC(1). Se muestran señales $\overline{\text{HSYNC}}$(2) y componente R(3).

FIGURA 5.8: Duración de un cuadro de video en la señal VSYNC(1). Se muestran señales $\overline{\text{HSYNC}}$(2) y componente R(3).

FIGURA 5.9: Ancho de pulso de la señal inactiva $\overline{\text{HSYNC}}$(2). Se muestran señales VSYNC(1) y componente R(3).

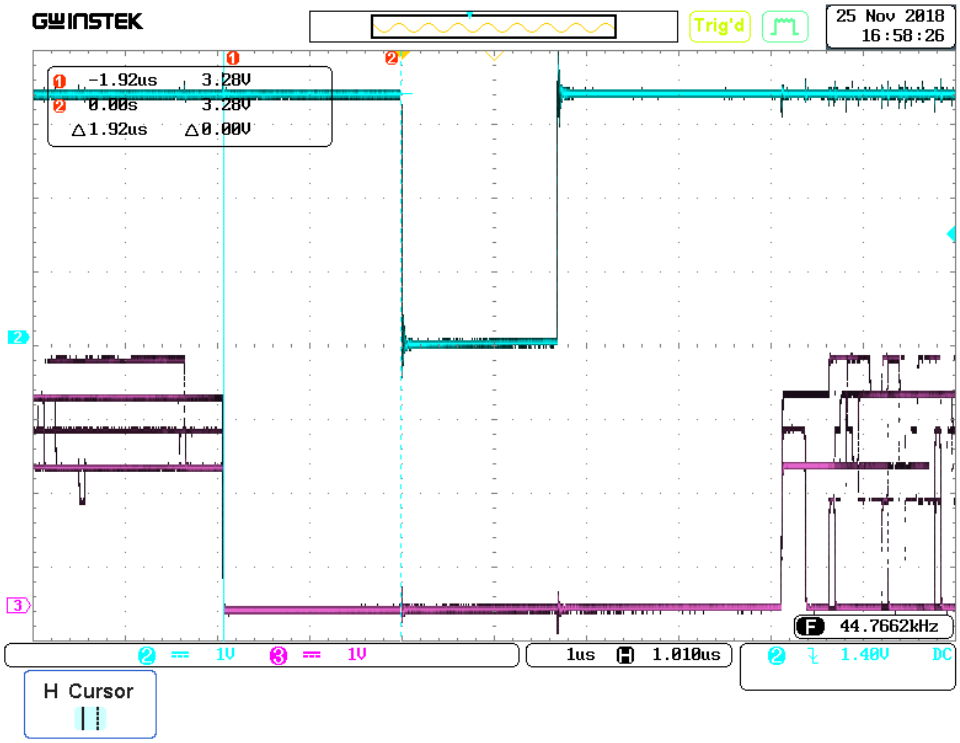


FIGURA 5.10: Horizontal Front Porch en señal $\overline{\text{HSYNC}}$(1). Se muestra señal de componente R(2).

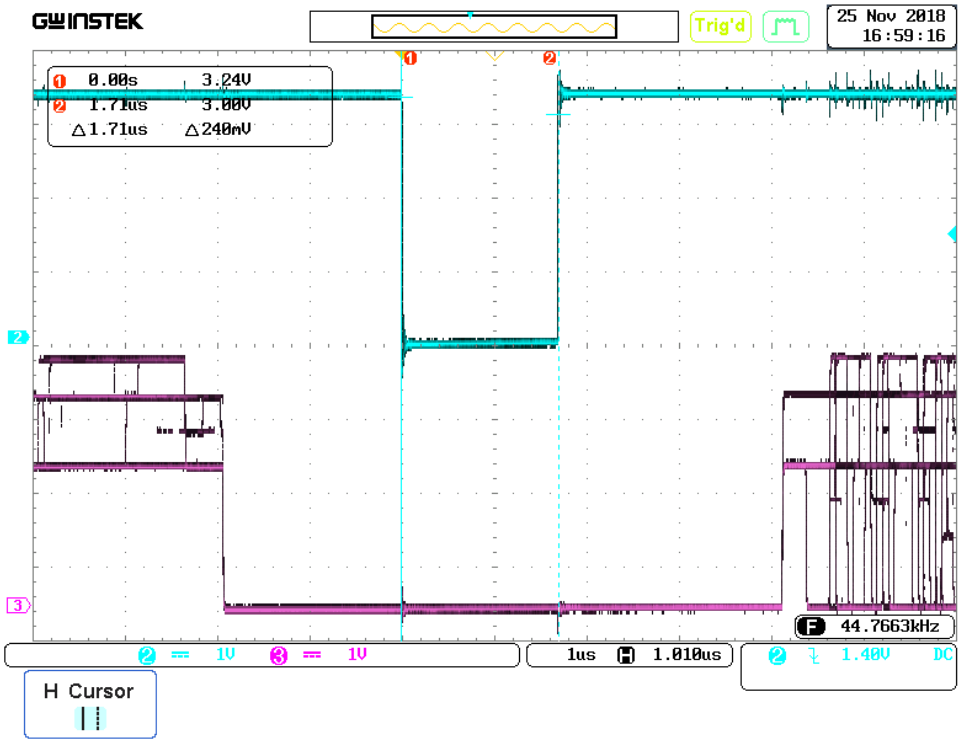


FIGURA 5.11: Ancho de pulso de la señal $\overline{\text{HSYNC}}$(1). Se muestra señal de componente R(2).

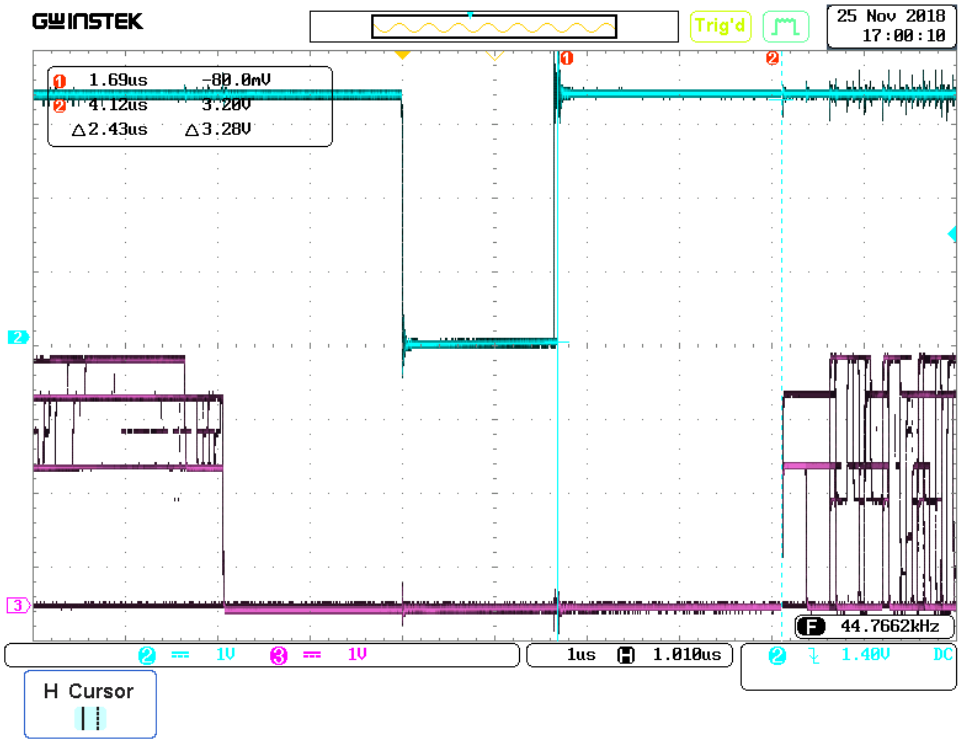


FIGURA 5.12: Horizontal Back Porch en señal $\overline{\text{HSYNC}}$(1). Se muestra señal de componente R(2).

Componentes de color

Se corrobora que el color de los pixeles en memoria se traduce correctamente en señales analógicas con y sin monitor conectado a la salida de video.

Para esto se generó un patrón de prueba en donde cada pixel incrementa de a un paso la intensidad de sus componentes de color. El código fuente se encuentra en el listado 5.1.

LISTADO 5.1: Código para generar patron de prueba de componentes de color

```
1  while (1)
2  {
3      DISPLAY_SwapBuffers ();
4      uint8_t * fb = DISPLAY_Get()->backBuffer;
5      for (uint32_t i = 0; i < 256 * 144; i += 8)
6      {
7          //          RRRGGGBB
8          fb[i  ] = 0b00000000;
9          fb[i+1] = 0b00100101;
10         fb[i+2] = 0b01001010;
11         fb[i+3] = 0b01101111;
12         fb[i+4] = 0b10010000;
13         fb[i+5] = 0b10110101;
14         fb[i+6] = 0b11011010;
15         fb[i+7] = 0b11111111;
16     }
17 }
```

El patrón de prueba generado puede apreciarse en la figura 5.13.

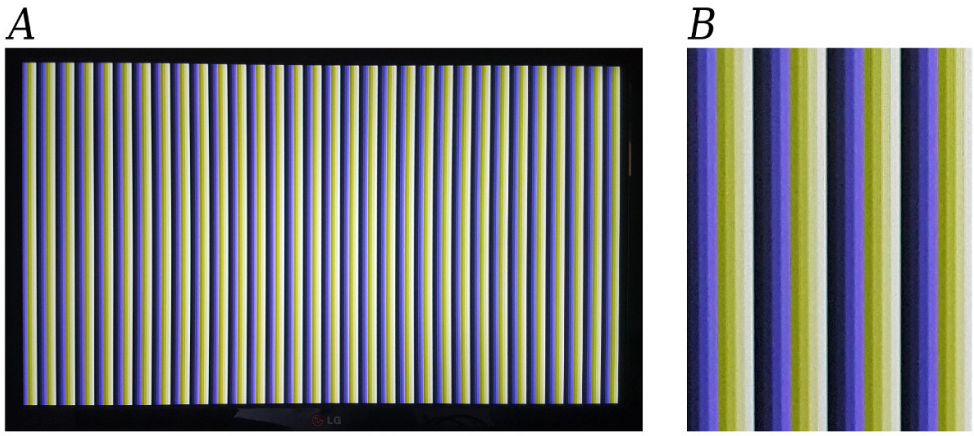


FIGURA 5.13: Patrón de prueba para componentes de color. (A) Foto de imagen real visualizada en un monitor. (B) Acercamiento de la foto para apreciar el patrón generado.

A continuación se utilizó un osciloscopio para visualizar las señales de salida. El trigger del osciloscopio fue configurado para sincronizarse el flanco de bajada de la señal HSYNC.

Las lecturas para rojo, verde y azul sin monitor conectado pueden verse en las figuras 5.14, 5.16 y 5.18 respectivamente.

El color azul tiene la mitad de niveles que rojo y verde. Por este motivo los valores se repiten dos veces por cada una de rojo y verde. Esto puede observarse comparando la figura 5.18 con 5.14 y 5.16.

Al conectar un monitor se esta conectando una resistencia de $75\,\Omega$ a masa y en paralelo con cada señal de color. En las figuras 5.15 5.17 y 5.19 se observan distorsiones notorias en las señales analógicas. Aun así, los niveles de tensión son correctos y la calidad de imagen es satisfactoria.

En la figura 5.14 se mide la duración de un pixel. La misma arroja una lectura de $64\,\mathrm{ns}$. Esa lectura es diferente del parámetro $TLP5 = 67{,}10\,\mathrm{ns}$ definido en la ecuación 3.7 ($4{,}6\,\%$ o $3{,}1\,\mathrm{ns}$ menor) y es consecuencia de decisiones en la implementación del driver de video.

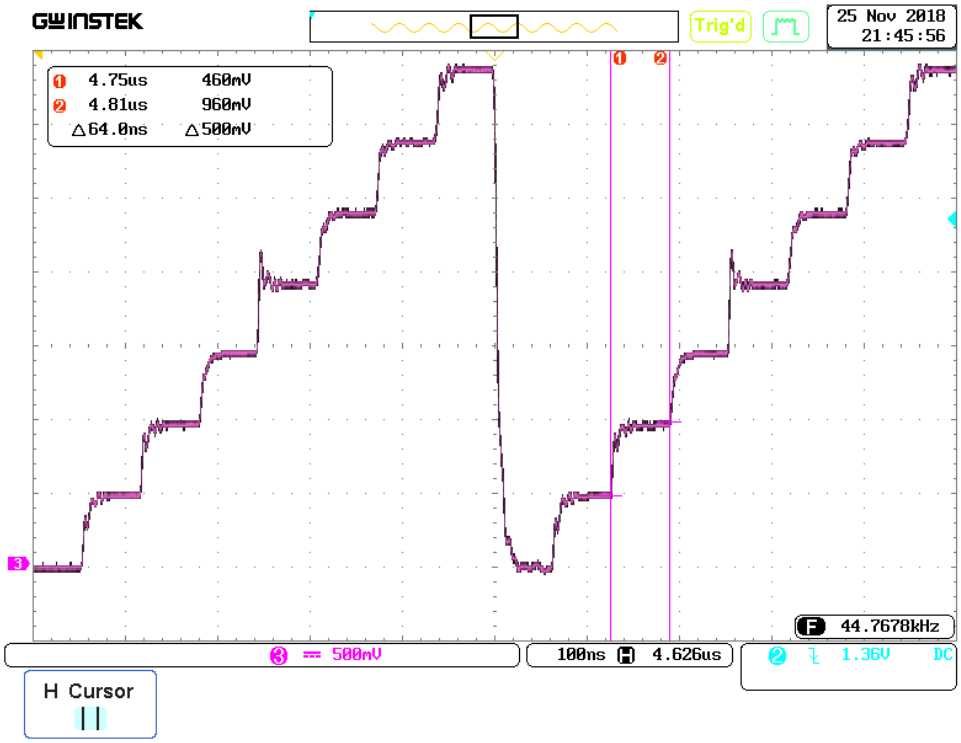


FIGURA 5.14: Niveles del componente rojo.

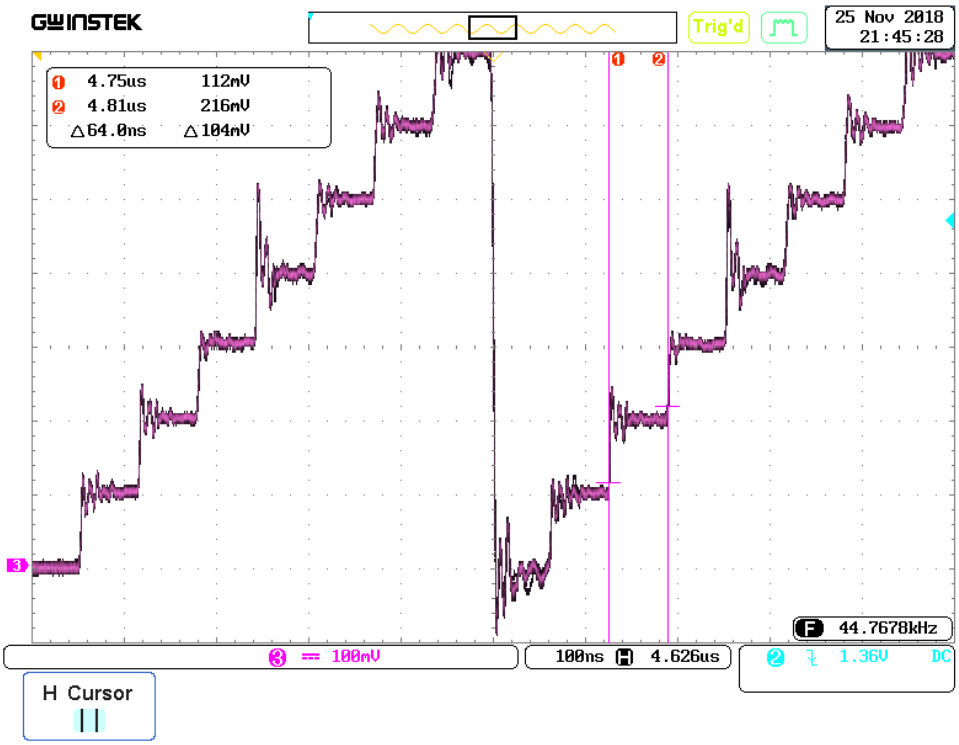


FIGURA 5.15: Niveles del componente rojo con monitor conectado.

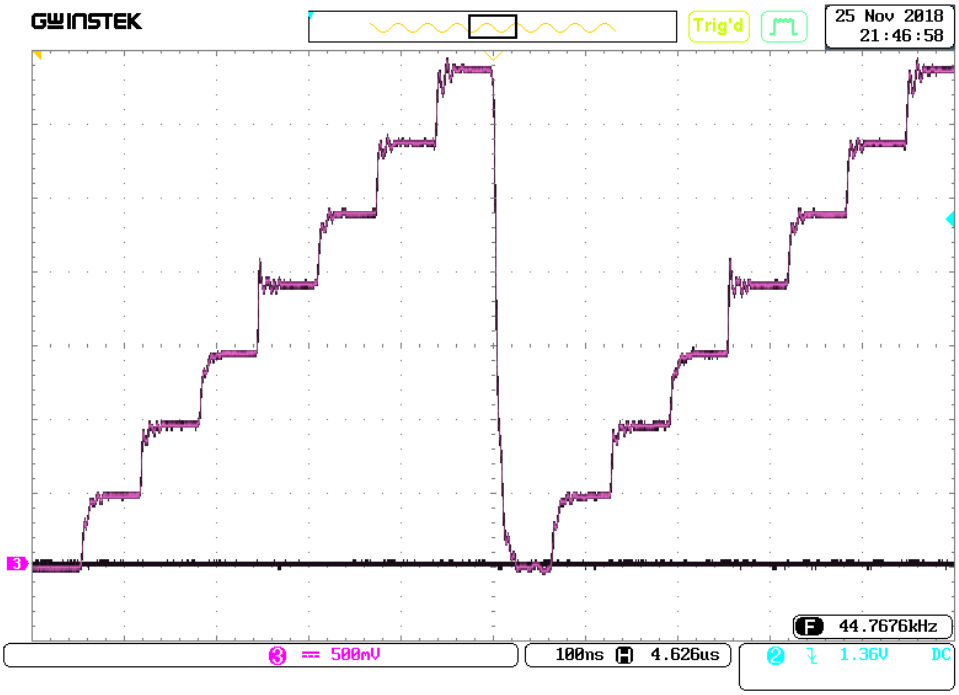


FIGURA 5.16: Niveles del componente verde.

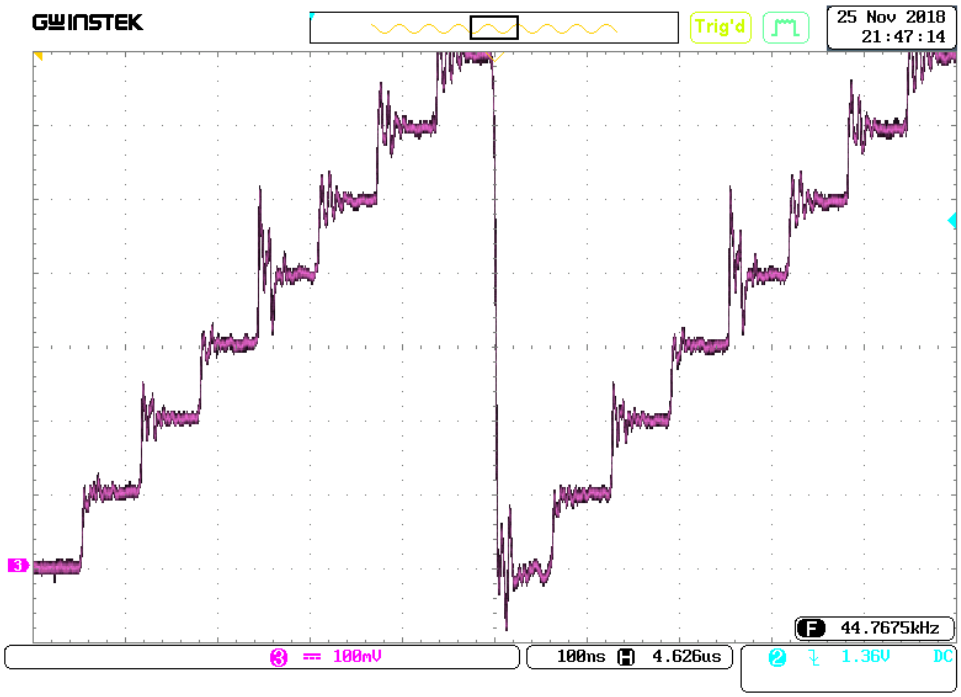


FIGURA 5.17: Niveles del componente verde con monitor conectado.

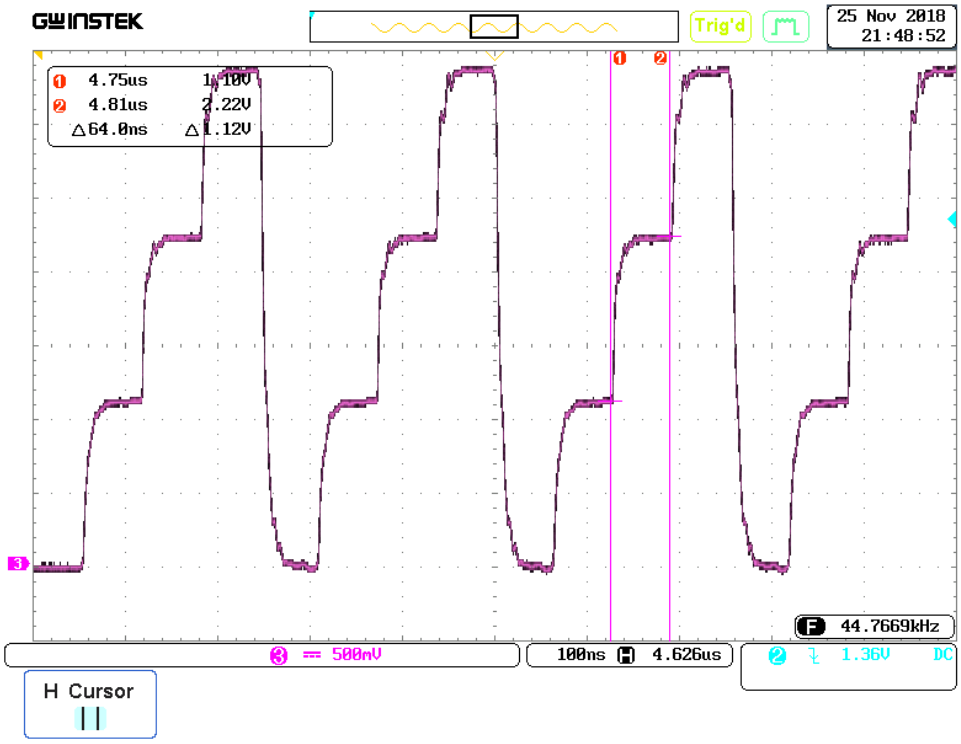


FIGURA 5.18: Niveles del componente azul.

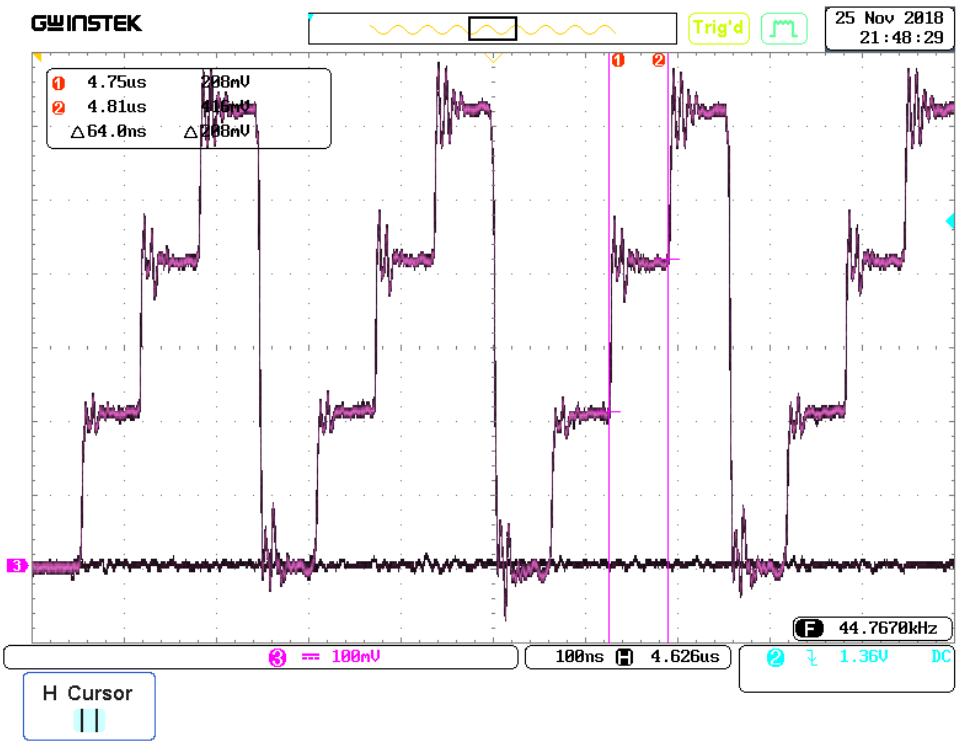


FIGURA 5.19: Niveles del componente azul con monitor conectado.

5.1.4. Calidad de imagen

Las pruebas se realizaron en diferentes momentos del desarrollo de firmware con diferentes monitores y televisores, con y sin adaptador VGA a HDMI. En todas la calidad de imagen fue similar al caso especifico que se comentará en esta subsección y que consiste en la conexión de la RETRO-CIAA a un televisor 4K Ultra HD.

En la figura 5.20 se observa una aplicación de prueba corriendo en dicho televisor. El video se ve fluido y el delineado de cada pixel se ve totalmente definido y sin ningún tipo de difuminado o «aguado» como el comentado en la subsección 3.2.7.

FIGURA 5.20: Calidad de imagen de la RETRO-CIAA en un televisor 4K Ultra HD.

La figura 5.21 es un acercamiento en donde se observa que el contraste de los colores es excelente y el tamaño de los pixeles es homogéneo.

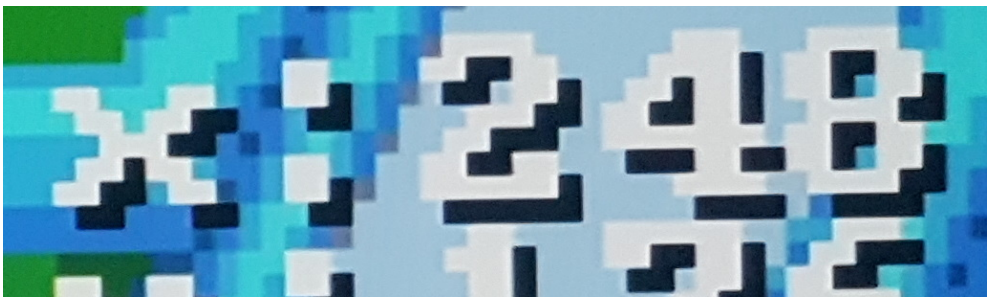


FIGURA 5.21: Calidad de imagen de la RETRO-CIAA. Acercamiento para notar detalles.

Finalmente, en la figura 5.22 puede verse el efecto de simulación de scanlines CRT configurado con dos intensidades. El brillo de la imagen a la derecha es levemente menor ya que es menor la cantidad de lineas activas. Este efecto se genera en tiempo real desde el driver de video con nulo impacto en la performance o el uso de memoria del sistema.

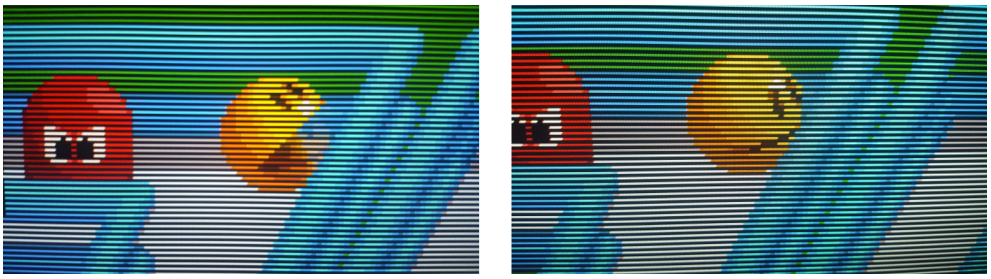


FIGURA 5.22: Efecto de scanlines de CRT en la RETRO-CIAA. A la izquierda: 5x3 pixeles con 2 scanlines. A la derecha: 5x2 pixeles con 3 scanlines.

Se concluye que fue cumplido el objetivo de lograr una excelente fluidez y calidad de imagen manteniendo la fidelidad del contenido de baja resolución en televisores modernos en donde a menor resolución de la señal, mayor es la distorsión que sufren las imágenes al ser escaladas a la resolución nativa de la pantalla.

También se observa que el efecto de scanlines CRT es efectivo en emular ese característica de la imagen en pantallas que no son CRT.

5.1.5. Salida de audio

Se utilizó un pendrive con musica en formato PCM signado de 16 bit y 32 kHz de frencuencia de muestreo. Luego se reprodujo en la salida de audio utilizando el código en el listado 5.2.

Lamentablemente no se dispone de equipamiento para realizar una prueba de distorsión armónica total con una señal conocida en la etapa de salida del audio.

Subjetivamente y a los hechos de comprobar la funcionalidad, se puede afirmar que el audio se escucha sin ninguna distorsión ni diferencia con otros dispositivos de audio.

LISTADO 5.2: Reproducción de archivo de música para prueba de audio

```
uint8_t col = 0;
while (1)
{
    DISPLAY_SwapBuffers (&app.display);
    DISPLAY_Clear       (&app.display, col);
    USBMS_Update        ();

    // Cambia color de fondo para indicar que aun esta vivo.
    if (DISPLAY_Get()->frameNumber % 8 == 0) ++ col;

    FONT_DrawText (&app.font, 8, 64, 0xFF, 0, "Prueba de música");

    // Se reproduce la música solo cuando el pendrive este conectado,
    // enumerado e iniciado.
    if (USBMS_Status() == USBMS_Status_StorageReadyMounted)
    {
        // Activa salida de audio y comienza a reproducir musica.
        SOUND_Mute      (false);
        SOUND_PlayBGM   ("USB:/bgm/musica.raw", SOUND_BGM_REPEAT_FOREVER);
    }
}
```

5.1.6. Señales de joystick

Se utilizó el osciloscopio para comprobar las señal de LATCH, CLOCK y la lectura de estado del joystick. El banco de pruebas utilizado puede verse en la figura 5.23.

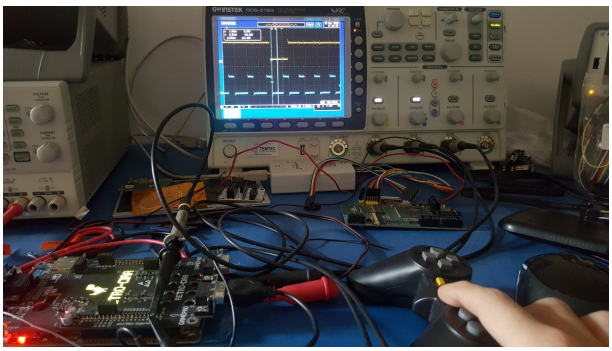


FIGURA 5.23: Banco de pruebas para comprobar señales de joystick.

Las señales de LATCH(1) y CLOCK(2) se observan en la figura 5.24.

En la figura 5.25 se realiza la medición del ancho de pulso de la señal de CLOCK(2) dando como resultado $248\,\text{ns}$.

En la misma señal de CLOCK(2) se realiza una medición del tiempo en estado bajo como se observa en la figura 5.26. Esta medición es variable según el pulso de CLOCK donde se la mida. En este caso el resultado de la medición es de $426\,\text{ns}$. En la implementación se utiliza el mismo tiempo de espera tanto para el ancho de pulso de CLOCK como para el tiempo de estado bajo. Inmediatamente después de la demora en estado bajo se procede a leer el joystick y luego se vuelve a generar un pulso de reloj. Esto significa que el tiempo de lectura del joystick $JDATA_{tRead}$ es igual a $426\,\text{ns} - 248\,\text{ns} = 178\,\text{ns}$. El valor estimado para $JDATA_{tRead}$ en la ecuación 3.9 fue de $78{,}43\,\text{ns}$. En futuros proyectos deberá ajustarse el método de dicha estimación a fin de lograr un valor mas representativo.

En la figura 5.27 se observa la lectura del estado del joystick 1 o DATA(1) en el tercer pulso de CLOCK(2). Como se observó en la figura 3.35, el valor bajo en DATA representa el estado presionado del botón de *START*. También se mide el tiempo de espera transcurrido desde que la señal se puso en estado bajo. Debido a que esa espera es la misma que el ancho de pulso de CLOCK, puede determinarse el momento aproximado en el cual se esta tomando la lectura de DATA(1).

Volviendo a la figura 5.24, la medición de tiempo entre el primer flanco de subida de LATCH(1) y el flanco de bajada del ultimo pulso de CLOCK(2) arrojó un resultado de $5{,}25\,\mu\text{s}$. Para obtener la medición del tiempo total del proceso de lectura $JSTAT_{tGet}$ hay que sumarle al ultimo resultado el de la medición de estado bajo de la figura 5.26. El resultado es $JSTAT_{tGet} = 5{,}25\,\mu\text{s} + 426\,\text{ns} = 5{,}676\,\mu\text{s}$. En la ecuación 3.10 este valor fue calculado como $3{,}827\,44\,\mu\text{s}$. Aun así, el valor leído se encuentra dentro de los margenes aceptables de diseño.

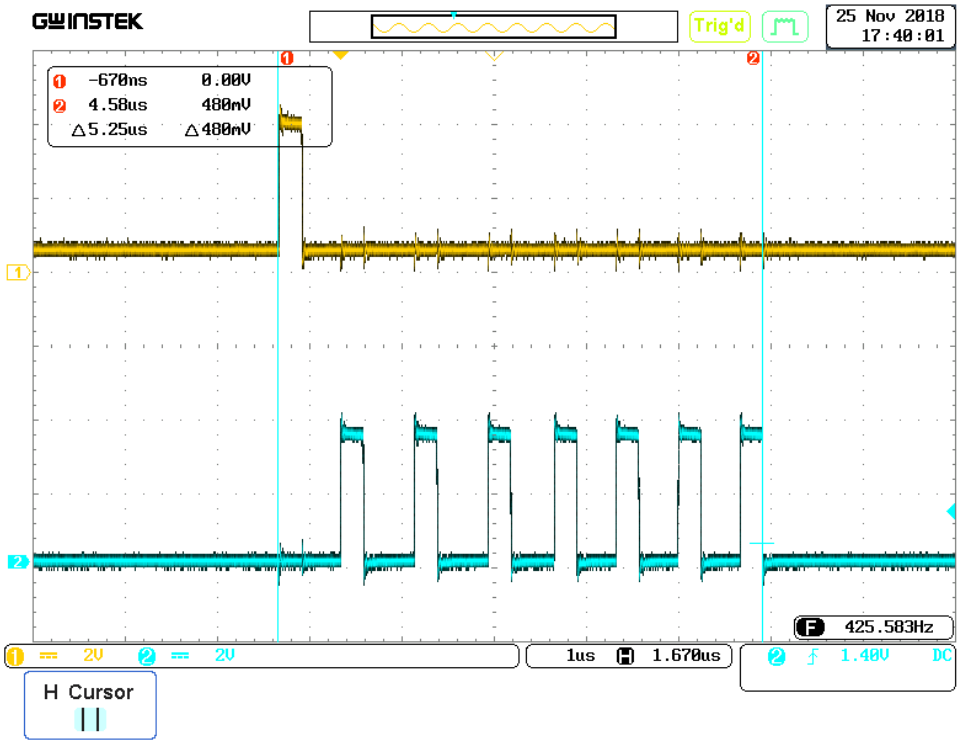


FIGURA 5.24: Señales de joystick para LATCH(1), CLOCK(2) y medición de tiempo total del proceso de lectura.

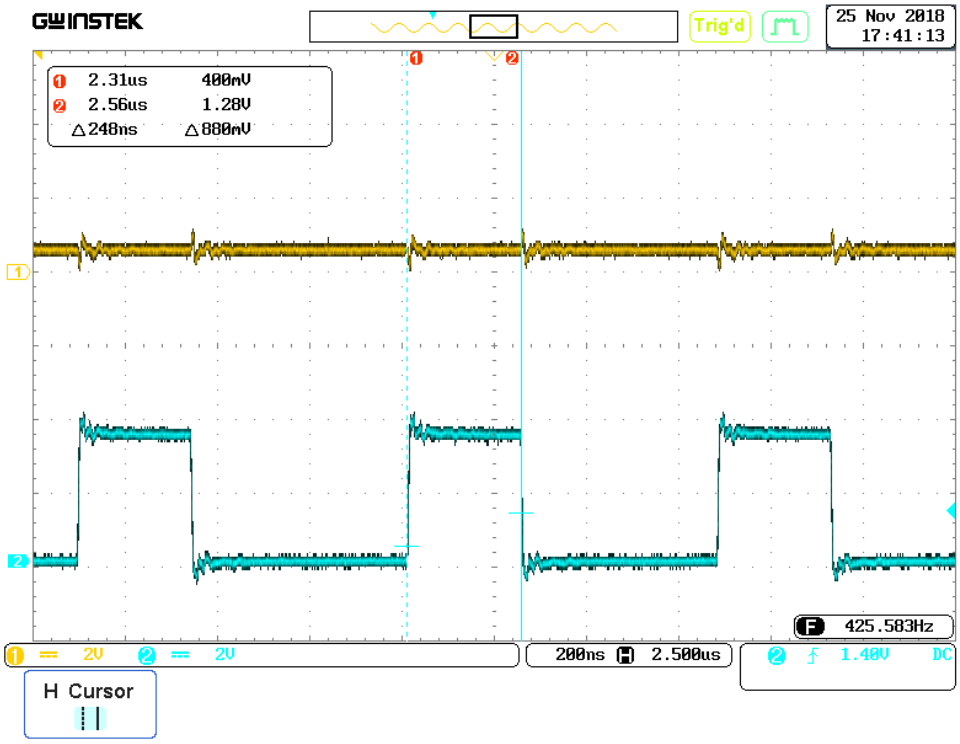


FIGURA 5.25: Medición del ancho de pulso de la señal de joystick para CLOCK(2).

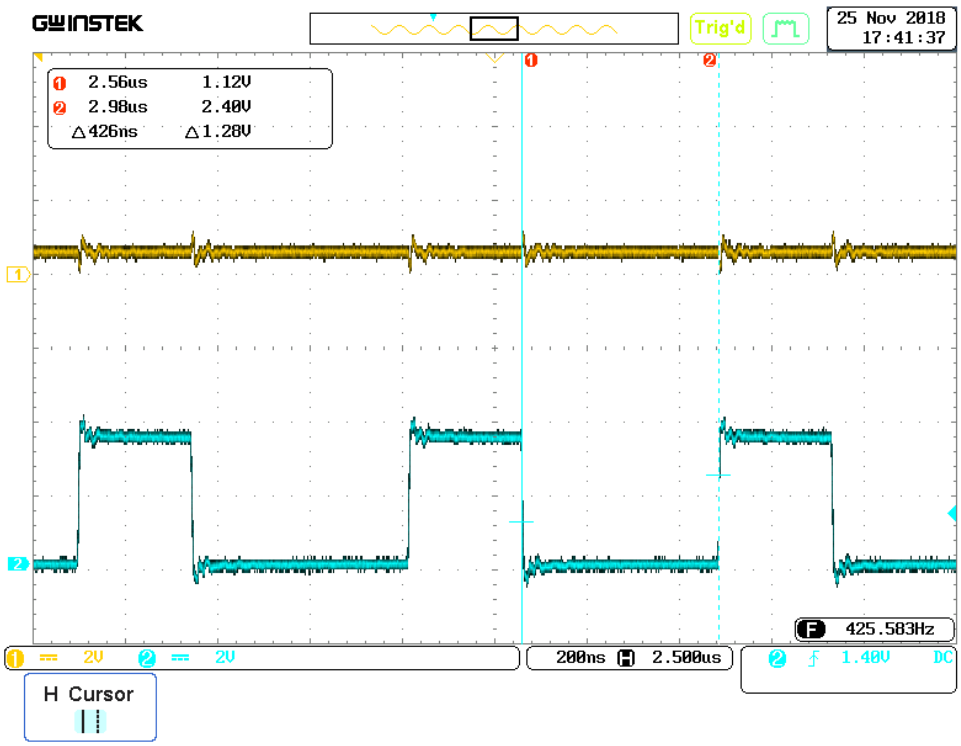


FIGURA 5.26: Medición de estado bajo de la señal de joystick para CLOCK(2).

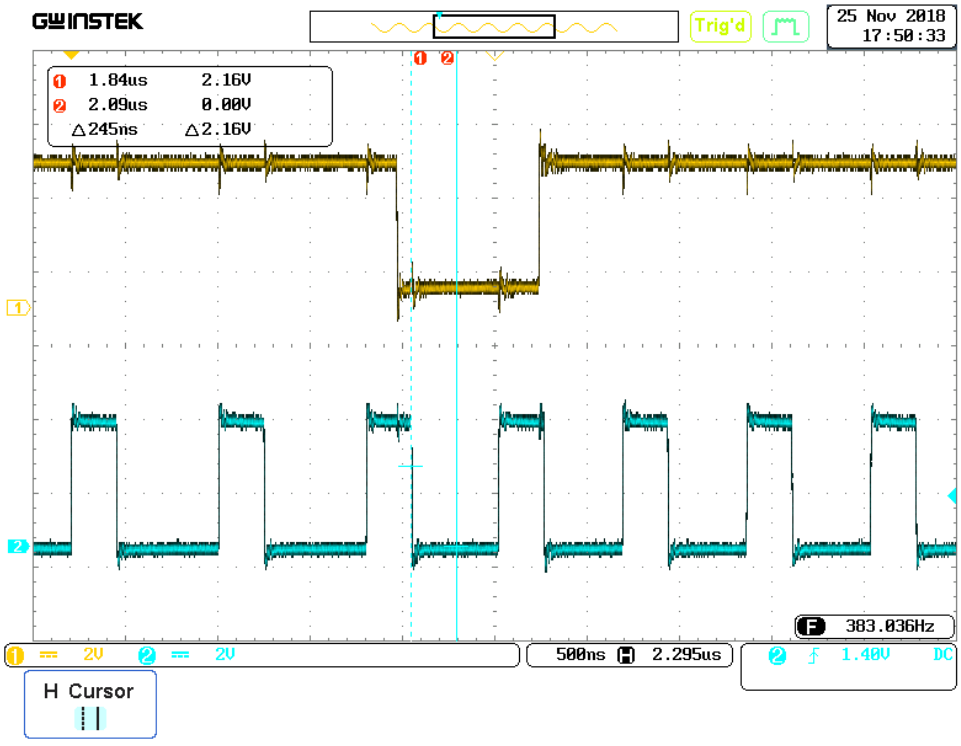


FIGURA 5.27: Estado de los datos del joystick (DATA(1)) y señal de CLOCK(2).

5.1.7. Modulo Wi-Fi

Se comprobó que el modulo Wi-Fi estuviese vivo leyendo las señales que emite dicho modulo en su UART de debug al inicio. En RETRO-CIAA esta señal se encuentra en la señal «TD» del puerto «WIFI UART». La lectura se realizó desde una PC mediante un adaptador USB-TTL. Según la hoja de datos del modulo [11] los datos leídos corresponden a su log de inicio.

En el listado 5.3 se copia el resultado de la lectura.

LISTADO 5.3: Resultado de la lectura del log de inicio del modulo Wi-Fi

```
I (I (9) boot: ESP-IDF v2.0-3-gbef9896 2nd stage bootloader
I (10) boot: compile time 05:59:45
I (10) boot: Enabling RNG early entropy source...
I (22) boot: SPI Speed      : 40MHz
I (35) boot: SPI Mode       : DIO
I (47) boot: SPI Flash Size : 4MB
I (60) boot: Partition Table:
I (71) boot: ## Label            Usage          Type ST Offset   Length
I (93) boot:  0 phy_init         RF data          01 01 0000f000 00001000
I (116) boot:  1 otadata          OTA data         01 00 00010000 00002000
I (140) boot:  2 nvs              WiFi data        01 02 00012000 0000e000
I (163) boot:  3 at_customize     unknown          40 00 00020000 000e0000
I (186) boot:  4 ota_0            OTA app          00 10 00100000 00180000
I (209) boot:  5 ota_1            OTA app          00 11 00280000 00180000
I (233) boot: End of partition table
I (246) boot: Disabling RNG early entropy source...
I (263) boot: Loading app partition at offset 00100000
I (1437) boot: segment 0: paddr=0x00100018 vaddr=0x00000000 size=0x0ffe8 ( 65512)
I (1438) boot: segment 1: paddr=0x00110008 vaddr=0x3f400010 size=0x1c5f0 (116208)
 map
I (1454) boot: segment 2: paddr=0x0012c600 vaddr=0x3ffb0000 size=0x0215c (  8540)
 load
I (1485) boot: segment 3: paddr=0x0012e764 vaddr=0x40080000 size=0x00400 (  1024)
 load
I (1508) boot: segment 4: paddr=0x0012eb6c vaddr=0x40080400 size=0x1b028 (110632)
 load
I (1587) boot: segment 5: paddr=0x00149b9c vaddr=0x400c0000 size=0x00034 (    52)
 load
I (1588) boot: segment 6: paddr=0x00149bd8 vaddr=0x00000000 size=0x06430 ( 25648)
I (1604) boot: segment 7: paddr=0x00150010 vaddr=0x400d0018 size=0x7a56c (501100)
 map
I (1631) heap_alloc_caps: Initializing. RAM available for dynamic allocation:
I (1654) heap_alloc_caps: At 3FFBA6B8 len 00025948 (150 KiB): DRAM
I (1675) heap_alloc_caps: At 3FFE8000 len 00018000 (96 KiB): D/IRAM
I (1696) heap_alloc_caps: At 4009B428 len 00004BD8 (18 KiB): IRAM
I (1717) cpu_start: Pro cpu up.
I (1729) cpu_start: Single core mode
I (1742) cpu_start: Pro cpu start user code
```

5.2. Pruebas de firmware

5.2.1. Pruebas estáticas

Se generó una prueba estática utilizando la herramienta CCCC (*a code counter for C and C++*) a fin de obtener métricas del código de la librería retro-ciaa. Se excluyen las librerías de soporte desarrolladas por el fabricante del microcontrolador. En el listado 5.4 se copia el comando utilizado para generar el reporte con CCCC.

LISTADO 5.4: Linea de comandos utilizada para generar un reporte del firmware con CCCC.

```
usuario@linux:$ cccc common/*.c common/*.h m0/*.c m0/*.h m4/audio/*.c m4/audio/*.
h m4/base/*.c m4/base/*.h m4/input/*.c m4/input/*.h m4/storage/*.c m4/storage/*.h
 m4/video/*.c m4/video/*.h
```

Los resultados se adjuntan como documentación en el apéndice D. No se realiza una valoración ya que aún no se han definido los parámetros deseables de dichas métricas.

Capítulo 6

Conclusiones

6.1. Conclusiones generales

La factibilidad de este trabajo fue lo primero en probarse y el diseño fue desarrollado al detalle intentando no dejar nada al azar, salvando toda restricción y aprovechando los recursos disponibles al máximo. Como consecuencia los resultados de este trabajo fueron acordes a las altas expectativas depositadas en los requerimientos y especificaciones.

Los logros técnicos pueden resumirse en lo siguiente:

- Utilización del núcleo Cortex-M0 en una aplicación de alta performance y temporizado preciso en la generación de señales de video HDTV.
- Utilización del núcleo Cortex-M4 para la generación de audio y video en tiempo real haciendo un uso intensivo del procesador y los diferentes tipos de memoria disponible.
- Un nuevo firmware para la EDU-CIAA-NXP + RETRO-CIAA orientado a los gráficos, la reproducción de sonido y la conectividad inalámbrica.

El aporte de este trabajo es brindarle a la EDU-CIAA-NXP la capacidad de generar contenido multimedia de alta calidad a través de una expansión de muy bajo costo.

6.2. Próximos pasos

- Con mas trabajo orientado a la producción de material didáctico y con la ayuda de la comunidad de sistemas embebidos, quizá se pueda lograr el objetivo original: fomentar el estudio de computación gráfica sobre una herramienta amigable y accesible.
- Se requiere mas trabajo en el firmware, en particular aplicaciones de prueba. Actualmente las capacidades técnicas de la plataforma están muy poco explotadas. Se estima que es posible desarrollar un renderer 3D con capacidad gráfica similar a la primer consola PlayStation.
- Las limitaciones de diseño de la EDU-CIAA-NXP justifican apuntar a una RETRO-CIAA autónoma basada en el mismo procesador LPC4337 pero mejorando cuestiones como la alimentación, el ruido, los planos de masa y el ruteo de señales.

Apéndice A

Esquemático

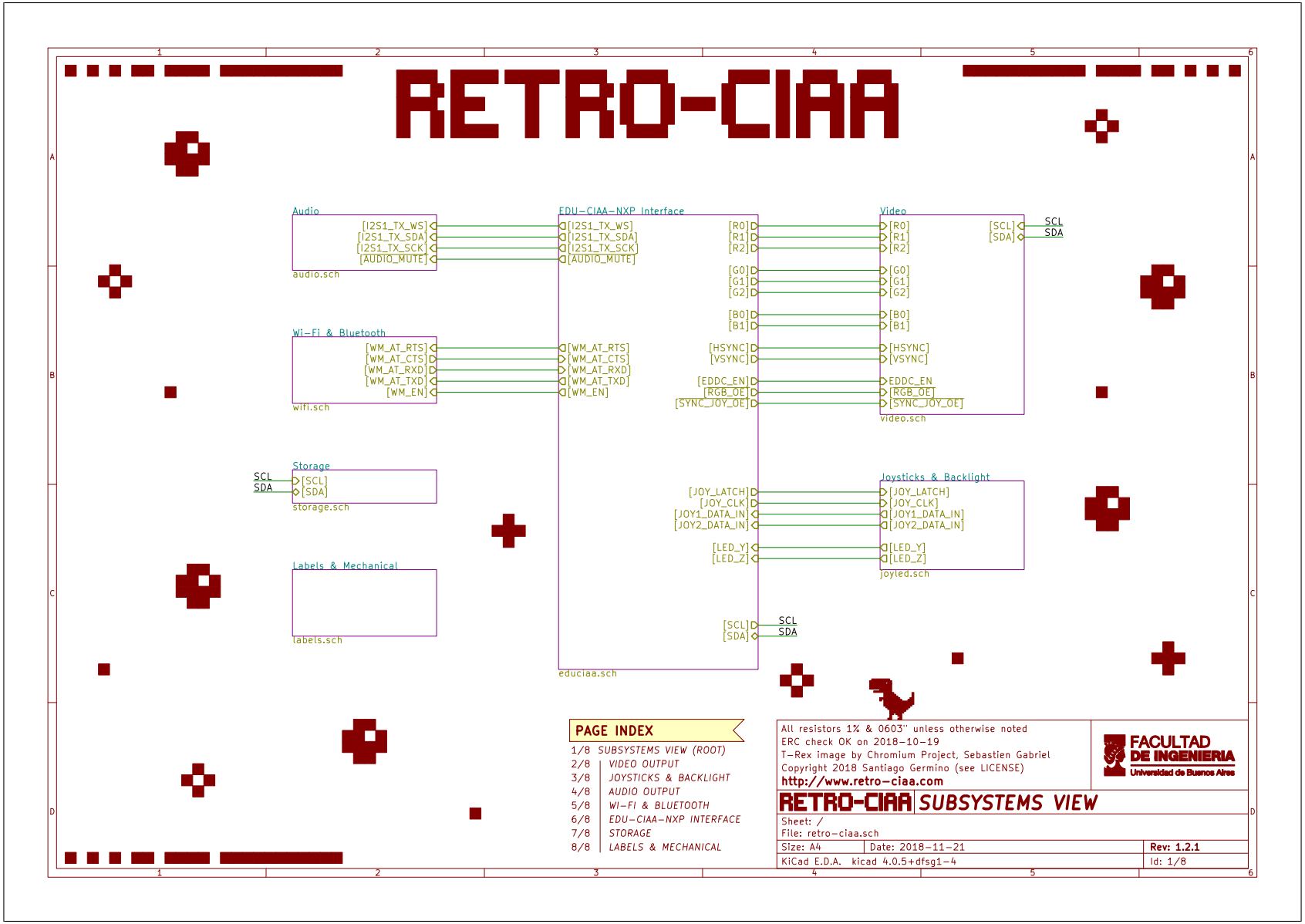


FIGURA A.1: Esquemático de RETRO-CIAA.

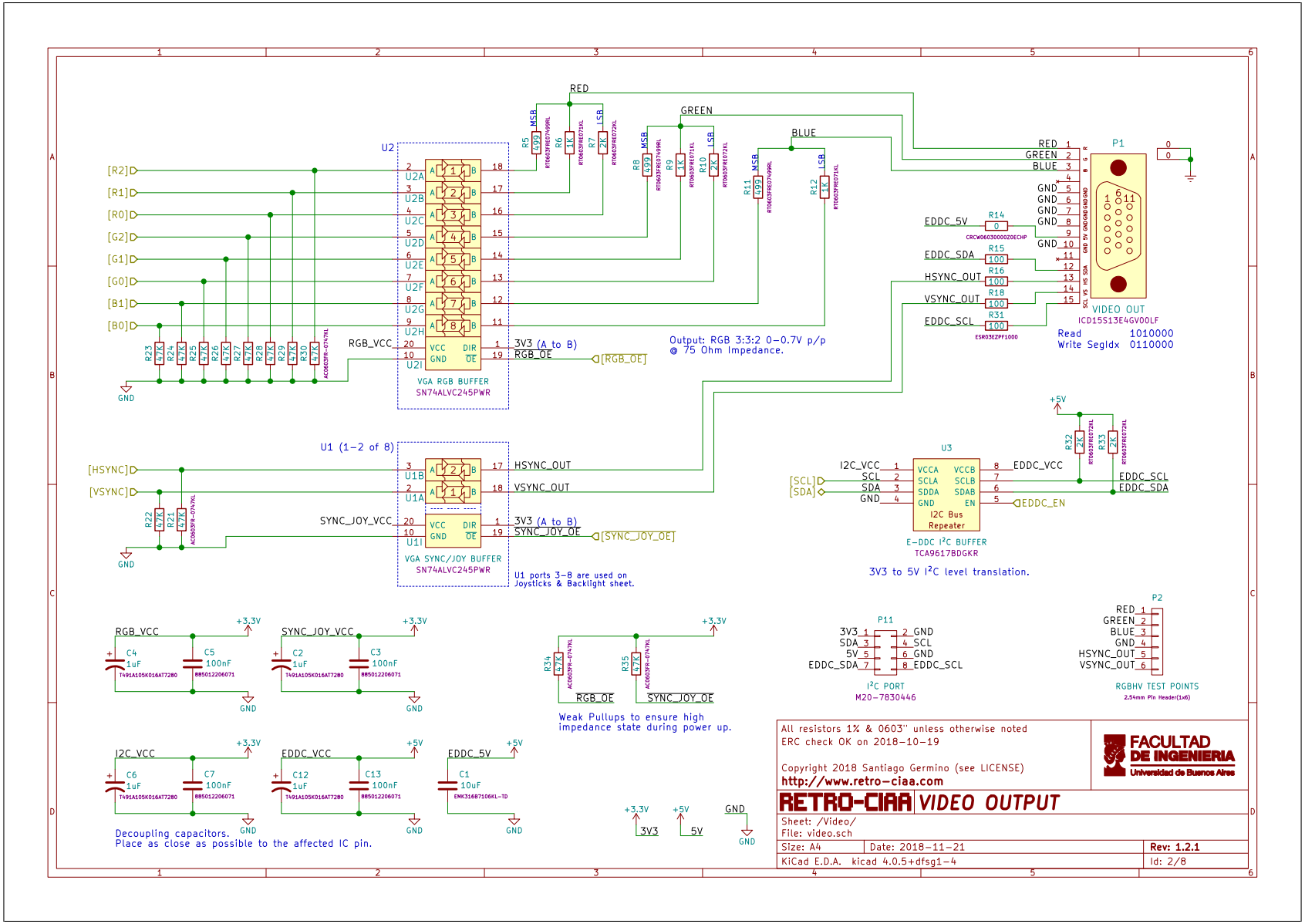


Figura A.1: Esquemático de RETRO-CIAA.

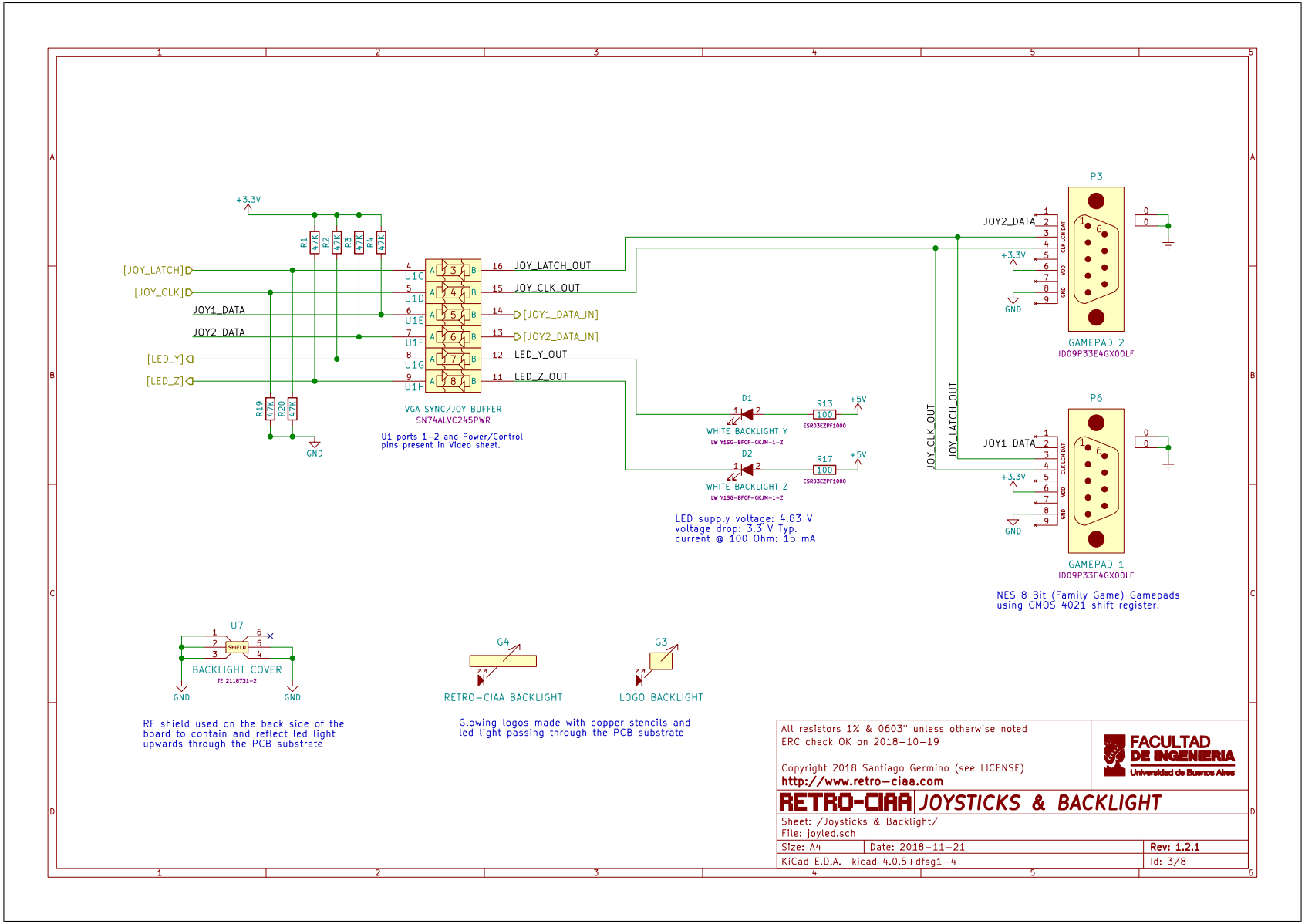


Figura A.1: Esquemático de RETRO-CIAA.

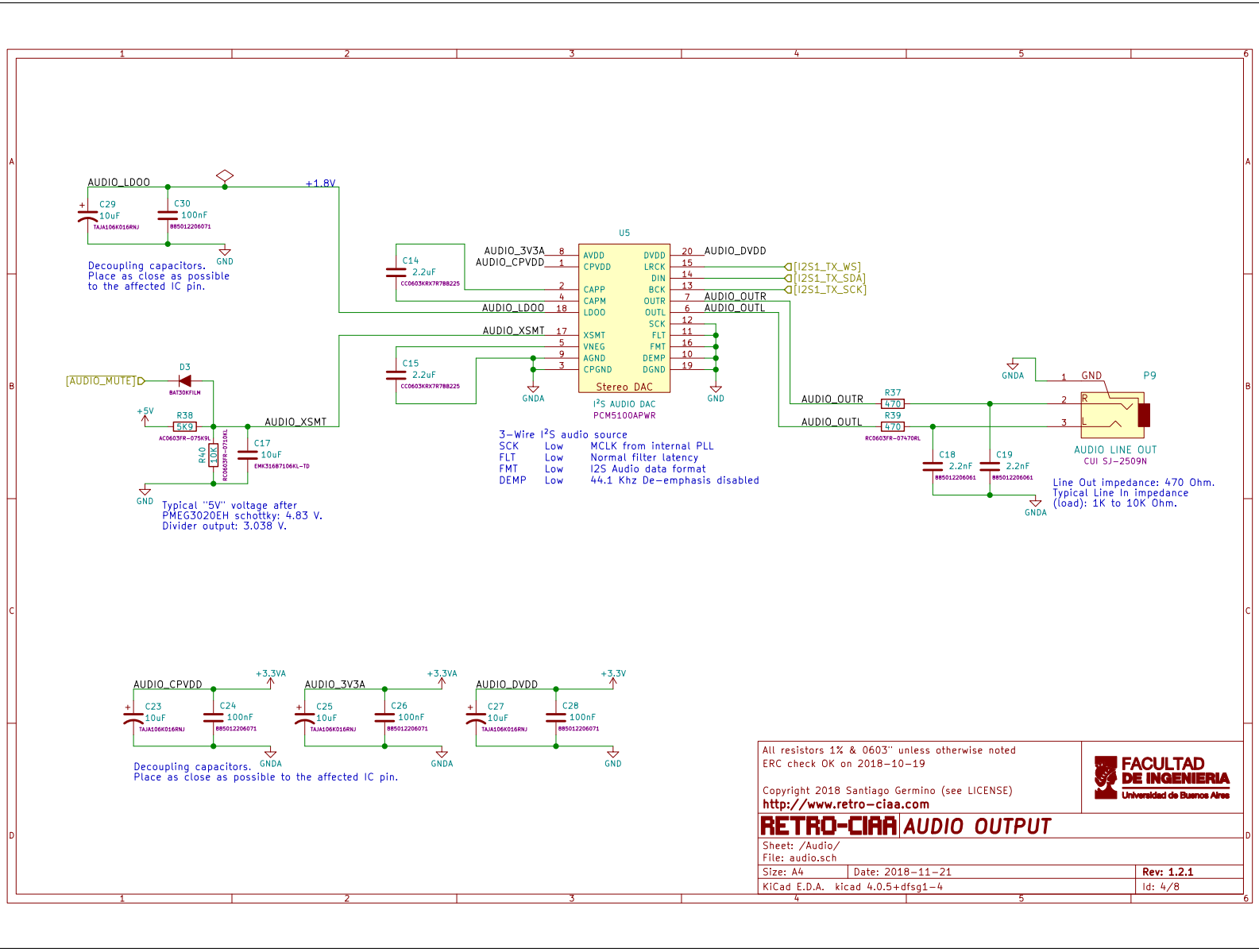


Figura A.1: Esquemático de RETRO-CIAA.

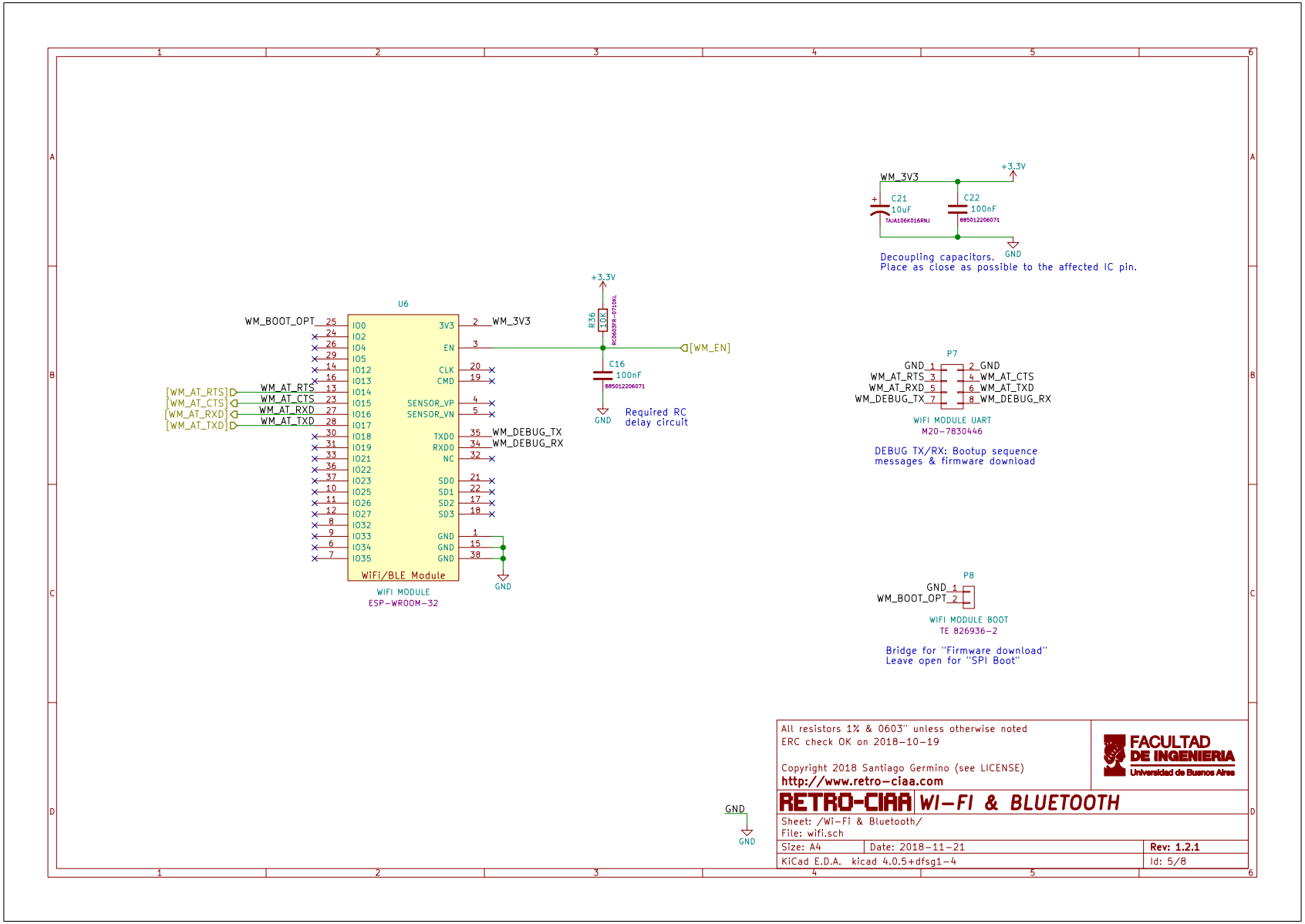


Figura A.1: Esquemático de RETRO-CIAA.

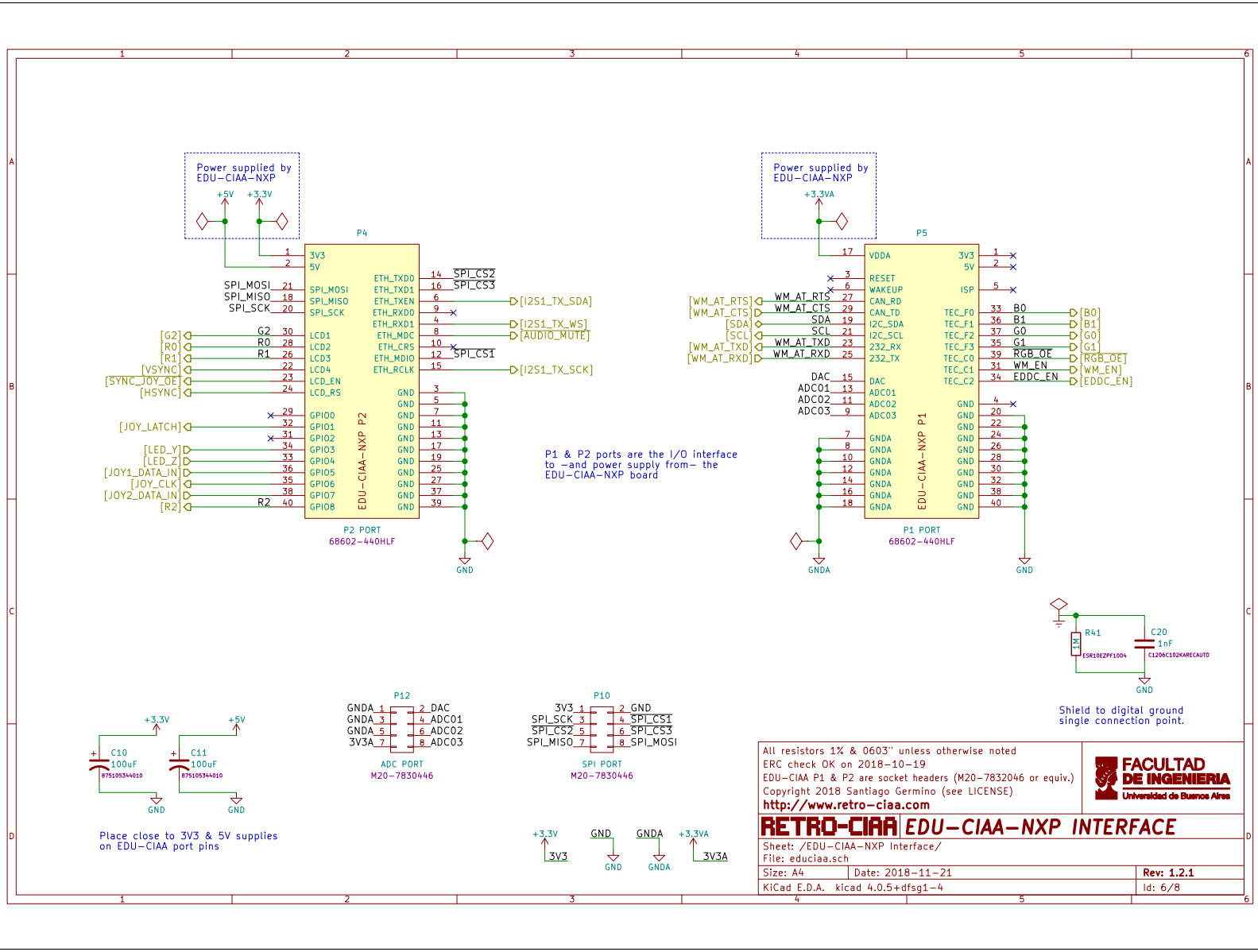


Figura A.1: Esquemático de RETRO-CIAA.

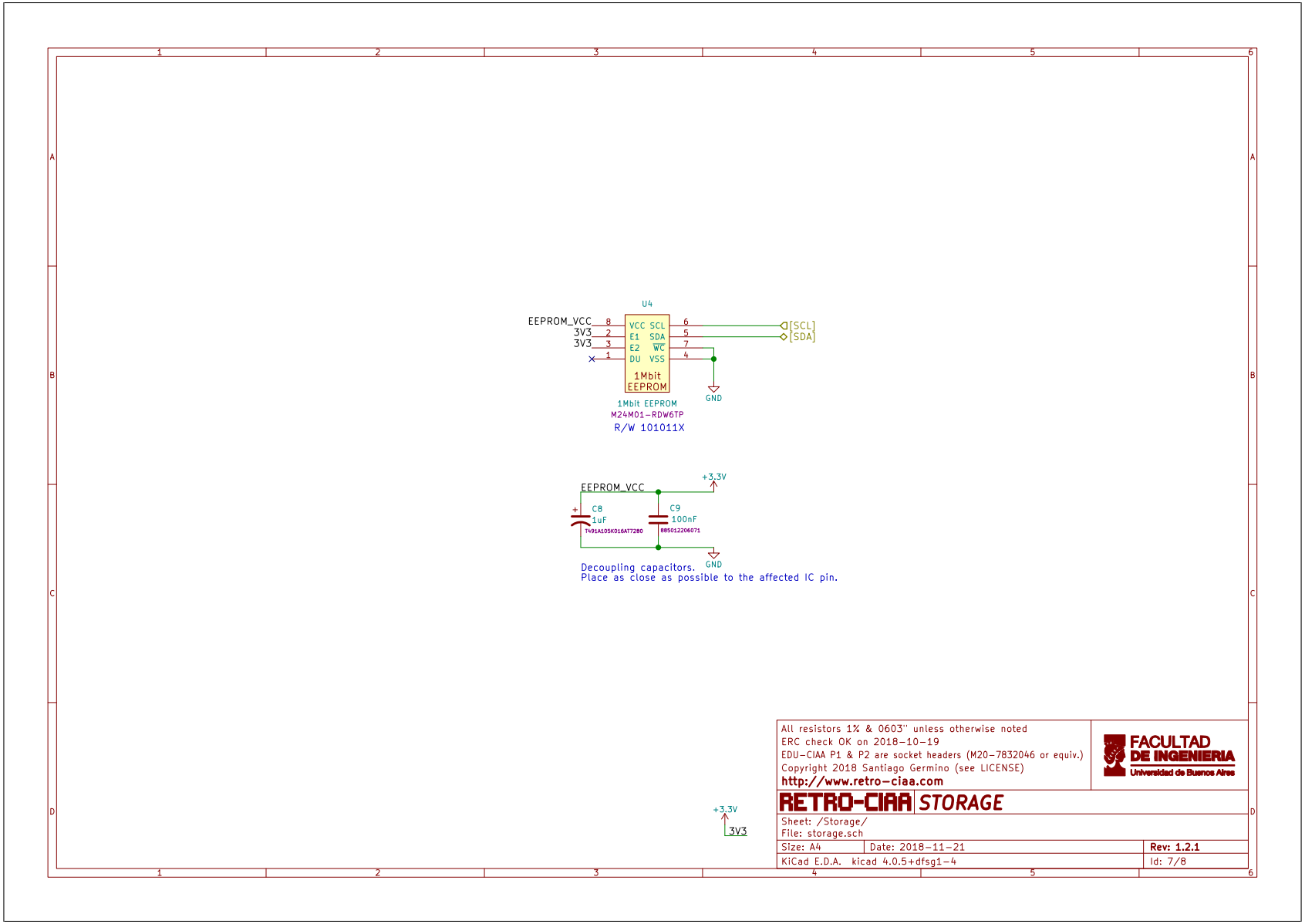


Figura A.1: Esquemático de RETRO-CIAA.

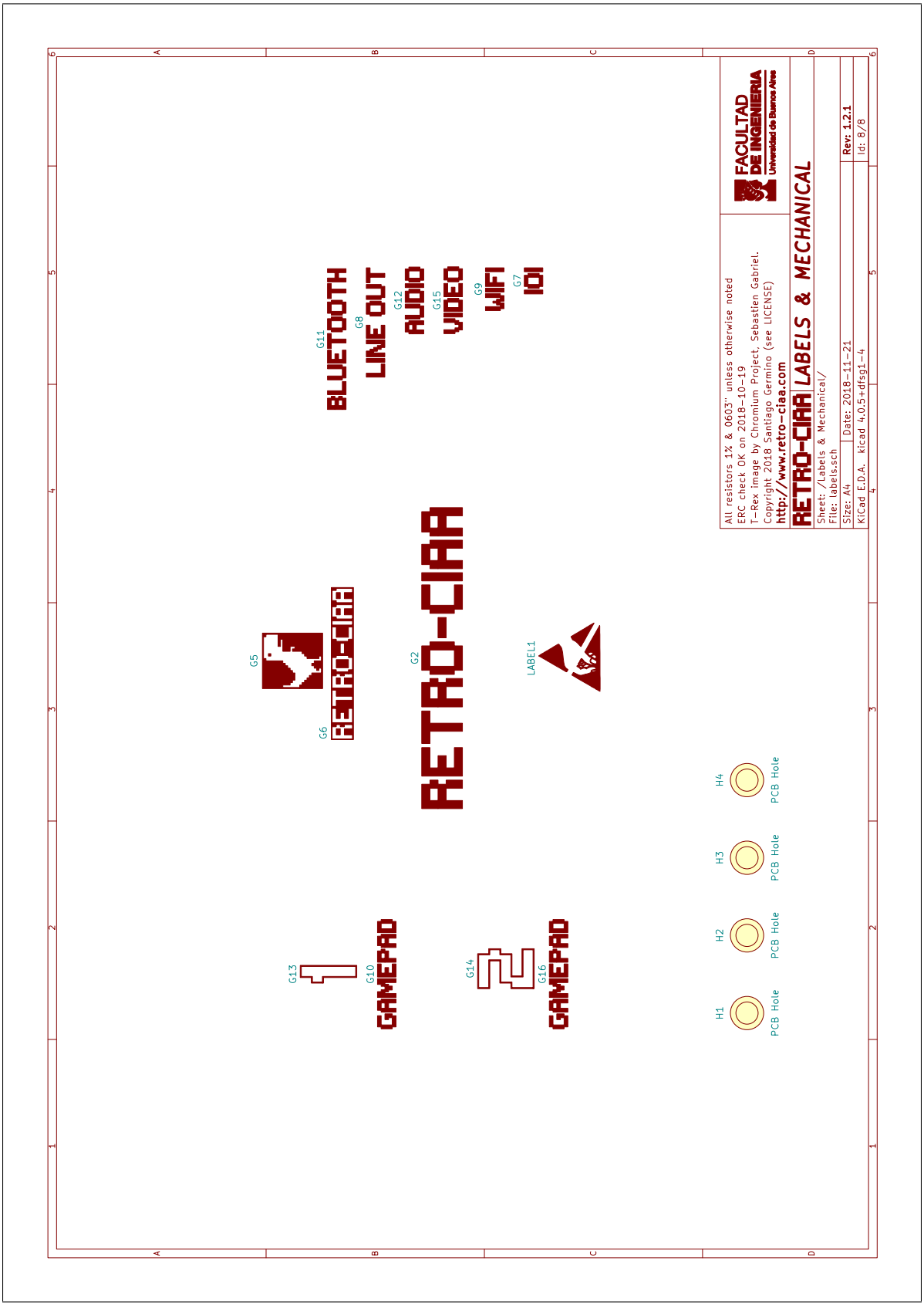


Figura A.1: Esquemático de RETRO-CIAA.

Apéndice B

Bill of materials

RETRO-CIAA BOM

RETRO-CIAA Bill of Materials

Proveedor: MOUSER **Precios en USD**

ID	Product Nr	Alternative	Annotate	Description	Case	Comment	Count	Price (M)	Min Qty	On Lite	Each PCB (	Each PCB (Lite)	Production (Full)
1	885012206071	VJ0603Y104JXQCW1BC (10V)		Multilayer Ceramic Capacitors MLCC - SMD/SMT WCAP-CSGP 100000pF 0603 10% 25V MLCC	0603"	100nF	11	$0.030	25	Y	$0.33	$0.33	
2	T491A105K016AT7280	293D105X9016A8T		Tantalum Capacitors - Solid SMD 16volts 1uF 10%	3216–18(A)	1uF	5	$0.194	10	Y	$0.97	$0.97	
3	875105344010			Aluminum Organic Polymer Capacitors WCAP-PSLP 16V 100uF 20% ESR=20mOhms	6.3x5.8 mm	100uF	2	$0.550	10	Y	$1.10	$1.10	
4	CC0603KRX7R7BB225	LMK107BJ225KAHT (X5R, 10V)		Multilayer Ceramic Capacitors MLCC - SMD/SMT 2.2uF 16V 10% Ceramic Capacitor	0603"	2.2uF	2	$0.354	10	Y	$0.71	$0.71	
5	885012206061			Multilayer Ceramic Capacitors MLCC - SMD/SMT WCAP-CSGP 2200pF 0603 10% 25V MLCC	0603"	2.2nF	2	$0.030	25	Y	$0.06	$0.06	
6	EMK316B7106KL-TD	GRM319C81C106KA12D (X6S)		Multilayer Ceramic Capacitors MLCC - SMD/SMT 10uF 16V X7R +/-10% 1206 Gen Purp	1206"	10uF	2	$0.275	10	Y	$0.55	$0.55	
7	TAJA106K016RNJ	F931C106KAA		Tantalum Capacitors - Solid SMD 16V 10uF 10% 1206 ESR= 3 Ohms	3216–18(A)	10uF	5	$0.254	10	Y	$1.27	$1.27	
8	C1206C102KARECAUTO			Multilayer Ceramic Capacitors MLCC - SMD/SMT 250V 1000pF X7R 1206 10% ESD	1206"	1nF	1	$0.101	10	Y	$0.10	$0.10	
9	ESR10EZPF1004			Thick Film Resistors - SMD 0805 1Mohm 1% Anti Surge AEC-Q200	0805"	1M	1	$0.142	10	Y	$0.14	$0.14	
10	RT0603FRE07499RL			Thin Film Resistors - SMD 1/10W 499 ohm 1% 50ppm	0603"	499, DAC	3	$0.046	10	Y	$0.14	$0.14	
11	RT0603FRE071KL			Thin Film Resistors - SMD 1K ohm 1% 1/10W	0603"	1K, DAC	3	$0.045	10	Y	$0.14	$0.14	
12	RT0603FRE072KL			Thin Film Resistors - SMD 1/10W 2K ohm 1% 50ppm	0603"	2K, DAC	4	$0.046	10	Y	$0.18	$0.18	
13	ESR03EZPF1000			Thick Film Resistors - SMD 0603 100ohm 1% Anti Surge AEC-Q200	0603"	100, 1/4W	6	$0.103	10	Y	$0.62	$0.62	
14	AC0603FR-0747KL			Thick Film Resistors - SMD 1/10W 47K ohm 1% AEC-Q200	0603"	47K	18	$0.006	100	Y	$0.11	$0.11	
15	RC0603FR-07470RL			Thick Film Resistors - SMD 470 OHM 1%	0603"	470	2	$0.009	10	Y	$0.02	$0.02	
16	AC0603FR-075K9L			Thick Film Resistors - SMD 1/10W 5.9K ohm 1% AEC-Q200	0603"	5K9	1	$0.016	10	Y	$0.02	$0.02	
17	RC0603FR-0710KL	AC0603FR-0710KL		Thick Film Resistors - SMD 10K OHM 1%	0603"	10K	2	$0.009	10	Y	$0.02	$0.02	
18	CRCW06030000Z0ECHP	RCS06030000Z0EA		Thick Film Resistors - SMD 1/4W Zeroohm Jumper High Power AEC-Q200	0603"	0 (Jumper)	1	$0.109	10		$0.11	$0.00	
19	LW Y1SG-BFCF-GKJM-1-Z			Standard LEDs - SMD White Diffused (Side View)	Custom	PCB Backlight	2	$0.790	10		$1.58	$0.00	
20	BAT30KFILM			Schottky Diodes & Rectifiers schottky diode	SOD-523		1	$0.301	10	Y	$0.30	$0.30	
21	SN74ALVC245PWR			Bus Transceivers Tri-State Octal Bus	TSSOP-20		2	$0.481	10	Y	$0.96	$0.96	
22	TCA9617BDGKR			Interface - Signal Buffers, Repeaters Level-Transl I2C Bus Repeater 8-VSSOP	VSSOP-8		1	$0.860	10		$0.86	$0.00	
23	M24M01-RDW6TP			EEPROM 1-Mbit I2C EEProm 256kB 1.7V to 5.5V	TSSOP-8		1	$1.330	10	Y	$1.33	$1.33	
24	PCM5100APWR			Audio D/A Converter ICs 2VRMS DirectPath 100 dB Audio Stereo DAC	TSSOP-20		1	$2.090	10	Y	$2.09	$2.09	
25	ESP-WROOM-32			WiFi / 802.11 Modules SMD Module, ESP32-D0WDQ6, 32Mbits SPI flash, UART Mode	Module		1	$3.800	1		$3.80	$0.00	
26	ICD15S13E4GV00LF			D-Sub High Density Connectors HD 15P RA SOCKET FM SCREW LCK UNC4.40		VGA	1	$1.310	1	Y	$1.31	$1.31	
27	ID09P33E4GX00LF			D-Sub Standard Connectors D-SUB 9 POSITION		Joysticks	2	$0.934	10	Y	$1.87	$1.87	
28	CUI SJ-2509N			Phone Connectors Audio Jacks	THT	Line out, 2.5mm	1	$0.600	1	Y	$0.60	$0.60	

Page 1

FIGURA B.1: RETRO-CIAA BOM (*Bill of Materials*).

RETRO-CIAA BOM

29	M20-7830446			Headers & Wire Housings 04+04 DIL VERTICAL SOCKET TIN	2.54 mm Socket Header (2x4)	WIFI / I2C / SPI / ADC Interfaces	4	$1.040	10		$4.16	$0.00	
30	68602-440HLF			Headers & Wire Housings 40P DR VT UNSHRD HDR TIN OVER NICKEL	2.54 mm Pin Header (2x40) Mating Post 5.84 mm	EDU-CIAA Interface (P1 & P2)	2	$0.874	10	Y	$1.75	$1.75	
31	TE 826936-2			Headers & Wire Housings 2P SINGLE ROW	2.54 mm Pin Header (1x2) Mating Post 5.8 mm	WIFI module boot option	1	$0.100	1		$0.10	$0.00	
32	TE 2118731-2			EMI Gaskets, Sheets & Absorbers CRS, 38.60mmx51.30mm Std Shield Cover		Backlight cover	1	$0.560	10		$0.56	$0.00	
33	PCB (ELECROW)			86mm x 145mm FR-4, Tickness 1.6mm, 2 Layers, Matte Black, HASL, Stencil. 4-7 Business Days			1	$3.734	20	Y	$3.73	$3.73	
34	PCB Shipping			Hong Kong DHL, 3-7 Business Days									$46.18
36	Components Shipping			UPS Standard Rate to Argentina									$35.00
										Total	$31.58	$20.41	

Page 2

Figura B.1: RETRO-CIAA BOM (*Bill of Materials*).

Apéndice C

Hojas de datos

LPC435x/3x/2x/1x

32-bit ARM Cortex-M4/M0 MCU; up to 1 MB flash and 136 kB SRAM; Ethernet, two High-speed USB, LCD, EMC

Rev. 5.3 — 15 March 2016 **Product data sheet**

1. General description

The LPC435X_3X_2X_1X are ARM Cortex-M4 based microcontrollers with Floating Point Unit (FPU) for embedded applications which include an ARM Cortex-M0 coprocessor, up to 1 MB of flash and 136 kB of on-chip SRAM, 16 kB of EEPROM memory, two high-speed USB controllers, Ethernet, LCD, an external memory controller, a quad SPI Flash Interface (SPIFI) that supports execute-in-place, advanced configurable peripherals such as the State Configurable Timer (SCTimer/PWM) and the Serial General Purpose I/O (SGPIO) interface, and multiple digital and analog peripherals. The LPC435X_3X_2X_1X operate at CPU frequencies of up to 204 MHz.

The ARM Cortex-M4 is a 32-bit core that offers system enhancements such as low power consumption, enhanced debug features, and a high level of support block integration. The ARM Cortex-M4 CPU incorporates a 3-stage pipeline, uses a Harvard architecture with separate local instruction and data buses as well as a third bus for peripherals, and includes an internal prefetch unit that supports speculative branching. The ARM Cortex-M4 supports single-cycle digital signal processing and SIMD instructions. A hardware floating-point processor is integrated into the core.

The ARM Cortex-M0 coprocessor is an energy-efficient and easy-to-use 32-bit core, which is upward code- and tool-compatible with the Cortex-M4 core. It is ideal for handling control or peripheral handling to free up the Cortex-M4 for real-time processing. The Cortex-M0 coprocessor offers up to 204 MHz performance with a simple instruction set and reduced code size. In LPC43xx, the Cortex-M0 coprocessor hardware multiply is implemented as a 32-cycle iterative multiplier.

For additional documentation related to the LPC43xx parts, see Section 17.

2. Features and benefits

- Cortex-M4 Processor core
  - ARM Cortex-M4 processor (version r0p1), running at frequencies of up to 204 MHz.
  - Built-in Memory Protection Unit (MPU) supporting eight regions.
  - Built-in Nested Vectored Interrupt Controller (NVIC).
  - Hardware floating-point unit.
  - Non-maskable Interrupt (NMI) input.
  - JTAG and Serial Wire Debug (SWD), serial trace, eight breakpoints, and four watch points.
  - Enhanced Trace Module (ETM) and Enhanced Trace Buffer (ETB) support.
  - System tick timer.

FIGURA C.1: Microcontrolador LPC4337JBD144.

- Cortex-M0 Processor core
  - ARM Cortex-M0 co-processor (version r0p0) capable of off-loading the main ARM Cortex-M4 application processor.
  - Running at frequencies of up to 204 MHz.
  - JTAG
  - Built-in NVIC.
- On-chip memory
  - Up to 1 MB on-chip dual bank flash memory with flash accelerator.
  - 16 kB on-chip EEPROM data memory.
  - 136 kB SRAM for code and data use.
  - Multiple SRAM blocks with separate bus access. Two SRAM blocks can be powered down individually.
  - 64 kB ROM containing boot code and on-chip software drivers.
  - 64 bit+ 256 bit of One-Time Programmable (OTP) memory for general-purpose use.
- Configurable digital peripherals
  - Serial GPIO (SGPIO) interface.
  - State Configurable Timer (SCTimer/PWM) subsystem on AHB.
  - Global Input Multiplexer Array (GIMA) allows to cross-connect multiple inputs and outputs to event driven peripherals like the timers, SCTimer/PWM, and ADC0/1.
- Serial interfaces
  - Quad SPI Flash Interface (SPIFI) with four lanes and up to 52 MB per second.
  - 10/100T Ethernet MAC with RMII and MII interfaces and DMA support for high throughput at low CPU load. Support for IEEE 1588 time stamping/advanced time stamping (IEEE 1588-2008 v2).
  - One High-speed USB 2.0 Host/Device/OTG interface with DMA support and on-chip high-speed PHY.
  - One High-speed USB 2.0 Host/Device interface with DMA support, on-chip full-speed PHY and ULPI interface to external high-speed PHY.
  - USB interface electrical test software included in ROM USB stack.
  - One 550 UART with DMA support and full modem interface.
  - Three 550 USARTs with DMA and synchronous mode support and a smart card interface conforming to ISO7816 specification. One USART with IrDA interface.
  - Up to two C_CAN 2.0B controllers with one channel each.
  - Two SSP controllers with FIFO and multi-protocol support. Both SSPs with DMA support.
  - One SPI controller.
  - One Fast-mode Plus I^2C-bus interface with monitor mode and with open-drain I/O pins conforming to the full I^2C-bus specification. Supports data rates of up to 1 Mbit/s.
  - One standard I^2C-bus interface with monitor mode and with standard I/O pins.
  - Two I^2S interfaces, each with DMA support and with one input and one output.
- Digital peripherals
  - External Memory Controller (EMC) supporting external SRAM, ROM, NOR flash, and SDRAM devices.

Figura C.1: Microcontrolador LPC4337JBD144.

- LCD controller with DMA support and a programmable display resolution of up to 1024 H × 768 V. Supports monochrome and color STN panels and TFT color panels; supports 1/2/4/8 bpp Color Look-Up Table (CLUT) and 16/24-bit direct pixel mapping. Available on parts LPC4357/53 only.
- Secure Digital Input Output (SD/MMC) card interface.
- Eight-channel General-Purpose DMA controller can access all memories on the AHB and all DMA-capable AHB slaves.
- Up to 164 General-Purpose Input/Output (GPIO) pins with configurable pull-up/pull-down resistors.
- GPIO registers are located on the AHB for fast access. GPIO ports have DMA support.
- Up to eight GPIO pins can be selected from all GPIO pins as edge and level sensitive interrupt sources.
- Two GPIO group interrupt modules enable an interrupt based on a programmable pattern of input states of a group of GPIO pins.
- Four general-purpose timer/counters with capture and match capabilities.
- One motor control Pulse Width Modulator (PWM) for three-phase motor control.
- One Quadrature Encoder Interface (QEI).
- Repetitive Interrupt timer (RI timer).
- Windowed watchdog timer (WWDT).
- Ultra-low power Real-Time Clock (RTC) on separate power domain with 256 bytes of battery powered backup registers.
- Alarm timer; can be battery powered.

- Analog peripherals
  - One 10-bit DAC with DMA support and a data conversion rate of 400 kSamples/s.
  - Two 10-bit ADCs with DMA support and a data conversion rate of 400 kSamples/s. Up to eight input channels per ADC.
- Unique ID for each device.
- Clock generation unit
  - Crystal oscillator with an operating range of 1 MHz to 25 MHz.
  - 12 MHz internal RC oscillator trimmed to 3 % accuracy over temperature and voltage (1.5 % accuracy for T_{amb} = 0 °C to 85 °C).
  - Ultra-low power Real-Time Clock (RTC) crystal oscillator.
  - Three PLLs allow CPU operation up to the maximum CPU rate without the need for a high-frequency crystal. The second PLL can be used with the High-speed USB, the third PLL can be used as audio PLL.
  - Clock output.
- Power
  - Single 3.3 V (2.4 V to 3.6 V) power supply with on-chip DC-to-DC converter for the core supply and the RTC power domain.
  - RTC power domain can be powered separately by a 3 V battery supply.
  - Four reduced power modes: Sleep, Deep-sleep, Power-down, and Deep power-down.
  - Processor wake-up from Sleep mode via wake-up interrupts from various peripherals.

Figura C.1: Microcontrolador LPC4337JBD144.

- Wake-up from Deep-sleep, Power-down, and Deep power-down modes via external interrupts and interrupts generated by battery powered blocks in the RTC power domain.
- Brownout detect with four separate thresholds for interrupt and forced reset.
- Power-On Reset (POR).

- Available as LQFP208, LQFP144, LBGA256, or TFBGA100 packages.

3. Applications

- Motor control
- Power management
- White goods
- RFID readers
- Embedded audio applications
- Industrial automation
- e-metering

Figura C.1: Microcontrolador LPC4337JBD144.

NCP1117, NCV1117

1.0 A Low-Dropout Positive Fixed and Adjustable Voltage Regulators

The NCP1117 series are low dropout positive voltage regulators that are capable of providing an output current that is in excess of 1.0 A with a maximum dropout voltage of 1.2 V at 800 mA over temperature. This series contains nine fixed output voltages of 1.5 V, 1.8 V, 1.9 V, 2.0 V, 2.5 V, 2.85 V, 3.3 V, 5.0 V, and 12 V that have no minimum load requirement to maintain regulation. Also included is an adjustable output version that can be programmed from 1.25 V to 18.8 V with two external resistors. On chip trimming adjusts the reference/output voltage to within ±1.0% accuracy. Internal protection features consist of output current limiting, safe operating area compensation, and thermal shutdown. The NCP1117 series can operate with up to 20 V input. Devices are available in SOT−223 and DPAK packages.

Features

- Output Current in Excess of 1.0 A
- 1.2 V Maximum Dropout Voltage at 800 mA Over Temperature
- Fixed Output Voltages of 1.5 V, 1.8 V, 1.9 V, 2.0 V, 2.5 V, 2.85 V, 3.3 V, 5.0 V, and 12 V
- Adjustable Output Voltage Option
- No Minimum Load Requirement for Fixed Voltage Output Devices
- Reference/Output Voltage Trimmed to ±1.0%
- Current Limit, Safe Operating and Thermal Shutdown Protection
- Operation to 20 V Input
- NCV Prefix for Automotive and Other Applications Requiring Unique Site and Control Change Requirements; AEC−Q100 Qualified and PPAP Capable
- These are Pb-Free Devices

Applications

- Consumer and Industrial Equipment Point of Regulation
- Active SCSI Termination for 2.85 V Version
- Switching Power Supply Post Regulation
- Hard Drive Controllers
- Battery Chargers

ON Semiconductor®

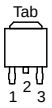
www.onsemi.com

SOT−223
ST SUFFIX
CASE 318H

DPAK
DT SUFFIX
CASE 369C

PIN CONFIGURATION

SOT−223
(Top View)



DPAK
(Top View)

Pin: 1. Adjust/Ground
2. Output
3. Input

Heatsink tab is connected to Pin 2.

ORDERING INFORMATION

See detailed ordering and shipping information in the package dimensions section on page 12 of this data sheet.

DEVICE MARKING INFORMATION

See general marking information in the device marking section on page 14 of this data sheet.

FIGURA C.2: Regulador 3,3 V NCP1117ST33T3G.

TEXAS INSTRUMENTS

PCM5100A, PCM5101A, PCM5102A
PCM5100A-Q1, PCM5101A-Q1, PCM5102A-Q1

SLAS859C – MAY 2012 – REVISED MAY 2015

PCM510xA 2.1 V_{RMS}, 112/106/100 dB Audio Stereo DAC with PLL and 32-bit, 384 kHz PCM Interface

1 Features

- Ultra Low Out-of-Band Noise
- Integrated High-Performance Audio PLL with BCK Reference to Generate SCK Internally
- Direct Line Level 2.1-V_{RMS} Output
- No DC Blocking Capacitors Required
- Line Level Output Down to 1KΩ
- Intelligent Muting System; Soft Up or Down Ramp and Analog Mute For 120-dB Mute SNR
- Accepts 16-, 24-, and 32-Bit Audio Data
- PCM Data Formats: I^2S, Left-Justified
- Automatic Power-Save Mode When LRCK And BCK Are Deactivated
- 1.8 V or 3.3 V Failsafe LVCMOS Digital Inputs
- Simple Configuration Using Hardware Pins
- Single-Supply Operation: 14
  - 3.3 V Analog, 1.8 V or 3.3 V Digital
- Qualified in Accordance with AEC-Q100

2 Applications

- A/V Receivers, DVD, BD Players
- Automotive Infotainment and Telematics
- HDTV Receivers
- Aftermarket Automotive Amplifiers

3 Description

The PCM510xA devices are a family of monolithic CMOS-integrated circuits that include a stereo digital-to-analog converter and additional support circuitry in a small TSSOP package. The PCM510xA devices use the latest generation of TI's advanced segment-DAC architecture to achieve excellent dynamic performance and improved tolerance to clock jitter.

Using Directpath™ charge-pump technology, the PCM510xA devices provide 2.1-V_{RMS} ground centered outputs, allowing designers to eliminate DC blocking capacitors on the output, as well as external muting circuits traditionally associated with single-supply line drivers.

The integrated line driver surpasses all other charge-pump based line drivers by supporting loads down to 1 kΩ per pin.

The integrated PLL on the device removes the requirement for a system clock (commonly known as master clock), allowing a 3-wire I^2S connection and reducing system EMI.

Intelligent clock error and PowerSense undervoltage protection utilizes a two-level mute system for pop-free performance.

Compared with many conventional switched capacitor DAC architectures, the PCM510xA family offers up to 20 dB lower out-of-band noise, reducing EMI and aliasing in downstream amplifiers/ADCs, measured from the traditional 100-kHz OBN measurements to 3 MHz).

Table 1. Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
PCM5102A	TSSOP (20)	5.50 mm × 4.40 mm
PCM5101A		
PCM5100A		

(1) For all available packages, see the orderable addendum at the end of the data sheet.

4 Simplified System Diagram

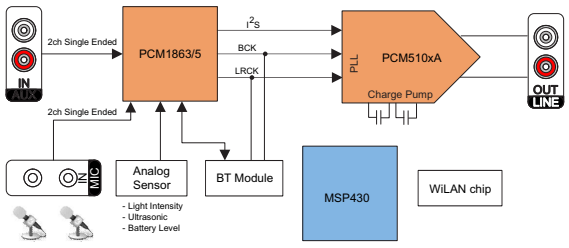


FIGURA C.3: DAC de audio estéreo PCM5100A.

TEXAS INSTRUMENTS
www.ti.com

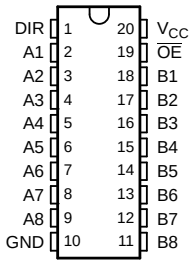
SN74ALVC245
OCTAL BUS TRANSCEIVER WITH 3-STATE OUTPUTS

SCES271D – APRIL 1999 – REVISED JULY 2004

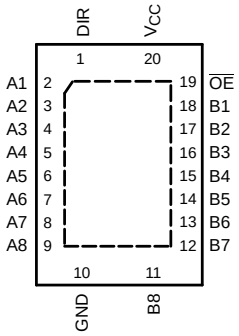
FEATURES

- **Operates from 1.65 V to 3.6 V**
- **Max t_{pd} of 3.4 ns at 3.3 V**
- **±24-mA Output Drive at 3.3 V**
- **Latch-Up Performance Exceeds 250 mA Per JESD 17**

DGV, DW, NS, OR PW PACKAGE
(TOP VIEW)



RGY PACKAGE
(TOP VIEW)



DESCRIPTION/ORDERING INFORMATION

This octal bus transceiver is designed for 1.65-V to 3.6-V V_{CC} operation.

The SN74ALVC245 is designed for asynchronous communication between data buses. The device transmits data from the A bus to the B bus or from the B bus to the A bus, depending on the logic level at the direction-control (DIR) input. The output-enable ($\overline{OE}$) input can be used to disable the device so the buses are effectively isolated.

To ensure the high-impedance state during power up or power down, $\overline{OE}$ should be tied to V_{CC} through a pullup resistor; the minimum value of the resistor is determined by the current-sinking capability of the driver.

ORDERING INFORMATION

T_A	PACKAGE(1)		ORDERABLE PART NUMBER	TOP-SIDE MARKING
-40°C to 85°C	QFN - RGY	Tape and reel	SN74ALVC245RGYR	VA245
	SOIC - DW	Tube	SN74ALVC245DW	ALVC245
		Tape and reel	SN74ALVC245DWR	
	SOP - NS	Tape and reel	SN74ALVC245NSR	ALVC245
	TSSOP - PW	Tube	SN74ALVC245PW	VA245
		Tape and reel	SN74ALVC245PWR	
	TVSOP - DGV	Tape and reel	SN74ALVC245DGVR	VA245

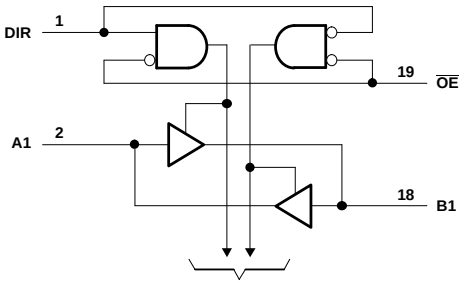
(1) Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

FIGURA C.4: Buffer de 8 puertos SN74ALVC245.

FUNCTION TABLE

INPUTS		OPERATION
$\overline{OE}$	DIR	
L	L	B data to A bus
L	H	A data to B bus
H	X	Isolation

LOGIC DIAGRAM (POSITIVE LOGIC)



ABSOLUTE MAXIMUM RATINGS(1)

over operating free-air temperature range (unless otherwise noted)

			MIN	MAX	UNIT
V_{CC}	Supply voltage range		-0.5	4.6	V
V_I	Input voltage range	Except I/O ports(2)	-0.5	4.6	V
		I/O ports(2)(3)	-0.5	V_{CC} + 0.5	
V_O	Output voltage range(2)(3)		-0.5	V_{CC} + 0.5	V
I_{IK}	Input clamp current	V_I < 0		-50	mA
I_{OK}	Output clamp current	V_O < 0		-50	mA
I_O	Continuous output current			±50	mA
	Continuous current through V_{CC} or GND			±100	mA
θ_{JA}	Package thermal impedance	DGV package(4)		92	°C/W
		DW package(4)		58	
		NS package(4)		60	
		PW package(4)		83	
		RGY package(5)		37	
T_{stg}	Storage temperature range		-65	150	°C

(1) Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) The input negative-voltage and output voltage ratings may be exceeded if the input and output current ratings are observed.
(3) This value is limited to 4.6 V, maximum.
(4) The package thermal impedance is calculated in accordance with JESD 51-7.
(5) The package thermal impedance is calculated in accordance with JESD 51-5.

Figura C.4: Buffer de 8 puertos SN74ALVC245.

TEXAS INSTRUMENTS

TCA9617B

SCPS259A – DECEMBER 2014 – REVISED DECEMBER 2014

TCA9617B Level-Translating FM+ I²C Bus Repeater

1 Features

- Two-Channel Bidirectional I²C Buffer
- Support for Standard Mode, Fast Mode (400 kHz), and Fast Mode+ (1 MHz) I²C Operation
- Operating Supply Voltage Range of 0.8 V to 5.5 V on A-Side
- Operating Supply Voltage Range of 2.2 V to 5.5 V on B-Side
- Voltage-Level Translation From 0.8 V to 5.5 V and 2.2 V to 5.5 V
- Footprint and Function Replacement for TCA9517
- Active-High Repeater-Enable Input
- Open-Drain I²C I/O
- 5.5-V Tolerant I²C and Enable Input Support
- Lockup-Free Operation
- Powered-Off High-Impedance I²C Bus Pins
- Support for Clock Stretching and Multiple Master Arbitration Across The Device
- Latch-Up Performance Exceeds 100 mA Per JESD 78, Class II
- ESD Protection Exceeds JESD 22
  - 4000-V Human-Body Model (A114-A)
  - 1500-V Charged-Device Model (C101)

2 Applications

- Servers
- Routers (Telecom Switching Equipment)
- Industrial Equipment
- Products With Many I²C Slaves and/or Long PCB Traces

3 Description

The TCA9617B is a BiCMOS dual bidirectional buffer intended for I²C bus and SMBus systems. It can provide bidirectional voltage-level translation (up-translation and down-translation) between low voltages (down to 0.8 V) and higher voltages (2.2 V to 5.5 V) in mixed-mode applications. This device enables I²C and similar bus systems to be extended, without degradation of performance even during level shifting.

The TCA9617B buffers both the serial data (SDA) and the serial clock (SCL) signals on the I²C bus, allowing two buses of 550 pF to be connected in an I²C application. This device can also be used to isolate two halves of a bus for voltage and capacitance.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
TCA9617B	VSSOP (8)	3.00 mm × 3.00 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

4 Simplified Schematic

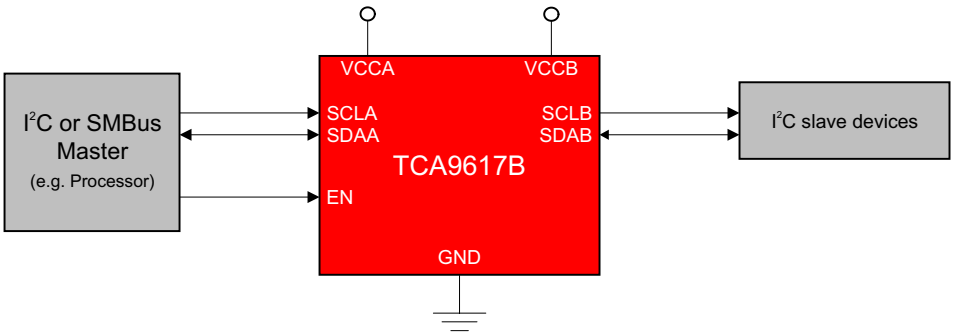


FIGURA C.5: Buffer I²C TCA9617B.

life.augmented

M24M01-R M24M01-DF

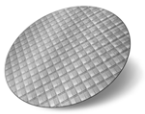
1-Mbit serial I²C bus EEPROM

Datasheet - production data

SO8 (MN)
150 mil width

TSSOP8 (DW)
169 mil width

WLCSP (CS, CX)



Unsawn wafer

Features

- Compatible with all I²C bus modes:
  - 1 MHz
  - 400 kHz
  - 100 kHz
- Memory array:
  - 1 Mbit (128 Kbyte) of EEPROM
  - Page size: 256 byte
  - Additional Write lockable page (M24M01-D order codes)
- Single supply voltage and high speed:
  - 1 MHz clock from 1.7 V to 5.5 V
- Write:
  - Byte Write within 5 ms
  - Page Write within 5 ms
- Operating temperature range:
  - from -40 °C up to +85 °C
- Random and sequential Read modes
- Write protect of the whole memory array
- Enhanced ESD/Latch-Up protection
- More than 4 million Write cycles
- More than 200-years data retention

Packages

- SO8 ECOPACK2®
- TSSOP8 ECOPACK2®
- WLCSP ECOPACK2®
- Unsawn wafer (each die is tested)

FIGURA C.6: EEPROM I²C de 1 Mbit M24M01.

1. Overview

ESP32-WROOM-32 (ESP-WROOM-32) is a powerful, generic Wi-Fi+BT+BLE MCU module that targets a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, music streaming and MP3 decoding.

At the core of this module is the ESP32-D0WDQ6 chip*. The chip embedded is designed to be scalable and adaptive. There are two CPU cores that can be individually controlled, and the clock frequency is adjustable from 80 MHz to 240 MHz. The user may also power off the CPU and make use of the low-power co-processor to constantly monitor the peripherals for changes or crossing of thresholds. ESP32 integrates a rich set of peripherals, ranging from capacitive touch sensors, Hall sensors, SD card interface, Ethernet, high-speed SPI, UART, I2S and I2C.

Note:

* For details on the part number of the ESP32 series, please refer to the document ESP32 Datasheet.

The integration of Bluetooth, Bluetooth LE and Wi-Fi ensures that a wide range of applications can be targeted, and that the module is future proof: using Wi-Fi allows a large physical range and direct connection to the internet through a Wi-Fi router, while using Bluetooth allows the user to conveniently connect to the phone or broadcast low energy beacons for its detection. The sleep current of the ESP32 chip is less than 5 μA, making it suitable for battery powered and wearable electronics applications. ESP32 supports a data rate of up to 150 Mbps, and 20.5 dBm output power at the antenna to ensure the widest physical range. As such the chip does offer industry-leading specifications and the best performance for electronic integration, range, power consumption, and connectivity.

The operating system chosen for ESP32 is freeRTOS with LwIP; TLS 1.2 with hardware acceleration is built in as well. Secure (encrypted) over the air (OTA) upgrade is also supported, so that developers can continually upgrade their products even after their release.

Table 1 provides the specifications of ESP32-WROOM-32 (ESP-WROOM-32).

Table 1: ESP32-WROOM-32 (ESP-WROOM-32) Specifications

Categories	Items	Specifications
Certification	RF certification	FCC/CE/IC/TELEC/KCC/SRRC/NCC
	Wi-Fi certification	Wi-Fi Alliance
	Bluetooth certification	BQB
	Green certification	RoHS/REACH
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 μs guard interval support
	Frequency range	2.4 GHz ~ 2.5 GHz
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and BLE specification
	Radio	NZIF receiver with -97 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH
	Audio	CVSD and SBC

FIGURA C.7: Módulo Wi-Fi y Bluetooth ESP-WROOM-32.

Categories	Items	Specifications
Hardware	Module interface	SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR
		GPIO, capacitive touch sensor, ADC, DAC
	On-chip sensor	Hall sensor, temperature sensor
	On-board clock	40 MHz crystal
	Operating voltage/Power supply	2.7 ~ 3.6V
	Operating current	Average: 80 mA
	Minimum current delivered by power supply	500 mA
	Operating temperature range	-40°C ~ +85°C
	Ambient temperature range	Normal temperature
	Package size	18±0.2 mm x 25.5±0.2 mm x 3.1±0.15 mm
Software	Wi-Fi mode	Station/SoftAP/SoftAP+Station/P2P
	Wi-Fi Security	WPA/WPA2/WPA2-Enterprise/WPS
	Encryption	AES/RSA/ECC/SHA
	Firmware upgrade	UART Download / OTA (download and write firmware via network or host)
	Software development	Supports Cloud Server Development / SDK for custom firmware development
	Network protocols	IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT
	User configuration	AT instruction set, cloud server, Android/iOS app

Figura C.7: Módulo Wi-Fi y Bluetooth ESP-WROOM-32.

Micro SIDELED 2808
Datasheet
Version 2.5

LW Y1SG

Micro SIDELED is a SMT LED with side emission. Due to its low package height it is ideal for applications in limited space environments.

Features:
- **Package:** white SMT package, colored diffused silicone resin
- **Technology:** InGaN on Sapphire
- **Viewing angle at 50 % I_V:** 120°
- **Color:** Cx = 0.30, Cy = 0.28 acc. to CIE 1931 (white); CTR = 8200 K
- **Optical efficiency (typ.):** 45 lm/W (white)

Applications
- Backlighting
- Coupling into light guides
- Signal and Symbol Luminary

Micro SIDELED ist eine SMT LED mit seitlicher Abstrahlrichtung. Aufgrund ihrer niedrigen Bauteilhöhe ist sie ideal für Anwendungen mit begrenztem Raum.

Besondere Merkmale:
- **Gehäusetyp:** weißes SMT Gehäuse, farbiger diffuser Silikon-Verguss
- **Technologie:** InGaN on Sapphire
- **Abstrahlwinkel bei 50 % I_V:** 120°
- **Farbe:** Cx = 0.30, Cy = 0.28 nach CIE 1931 (weiß); CTR = 8200 K
- **Optischer Wirkungsgrad (typ.):** 45 lm/W (weiß)

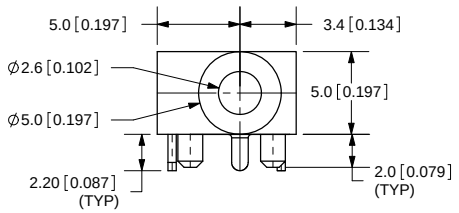
Anwendungen
- Hinterleuchtung
- Einkopplung in Lichtleiter
- Signal- und Symbolleuchten

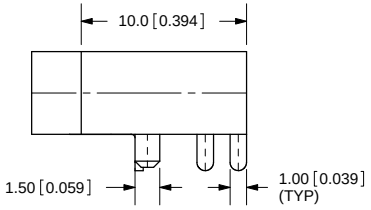
OSRAM
Opto Semiconductors

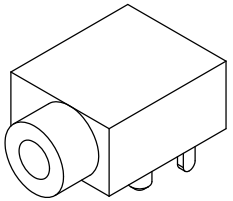
FIGURA C.8: LED luz blanca de emisión lateral LW Y1SG.

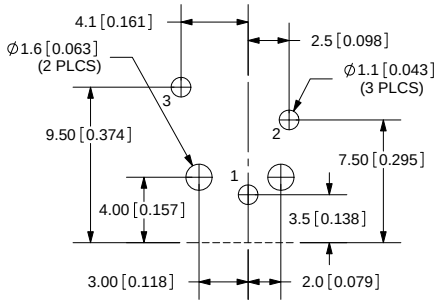
For more information, please visit the product page.

REV.	DESCRIPTION	DATE
A	NEW DRAWING	11/30/2005
B	added missing dimension	9/28/2009

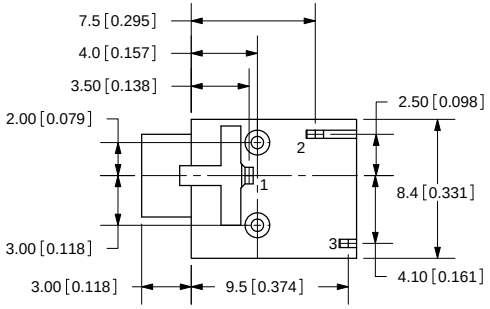








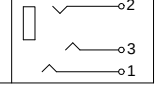
PCB LAYOUT
TOP VIEW



TOLERANCE:
X. ±0.5mm
X.X ±0.3mm
X.XX ±0.15mm

RoHS

SPECIFICATIONS:
RATING: 12V DC @ 1A
CONTACT RESISTANCE: 30m OHMS MAX
INSULATION RESISTANCE: 100M OHMS MIN: 500V DC
VOLTAGE WITHSTAND: 500V AC R.M.S. FOR 1 MINUTE
LIFE: 5,000 CYCLES
OPERATING TEMP: -20° ~ +60°C

MODEL NO.	SJ-2509N
SCHEMATIC	

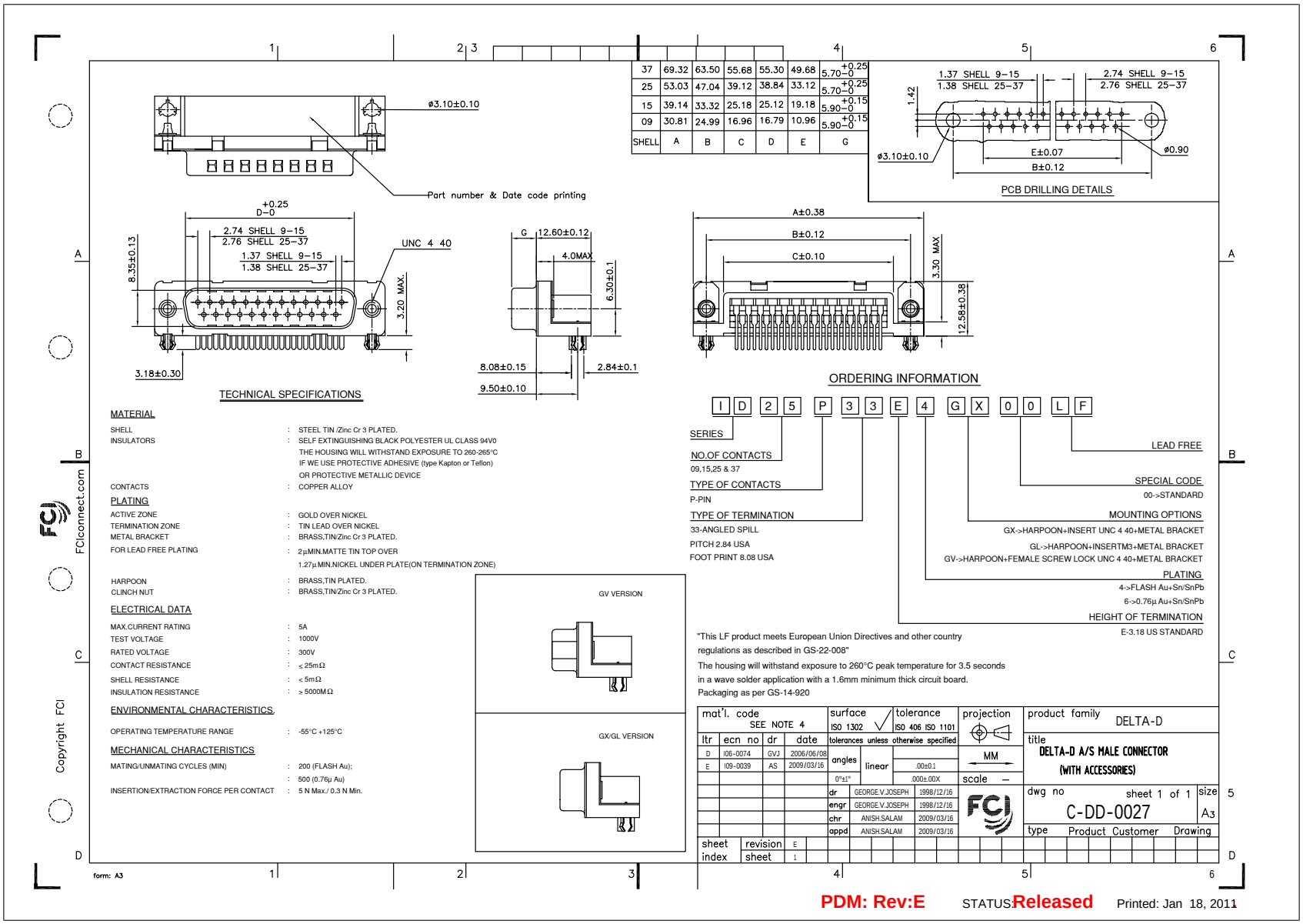
	MATERIAL	PLATING
TERMINAL 1	PBS	Tin
TERMINAL 2	Brass	Tin
TERMINAL 3	PBS	Tin
HOUSING	PBT	

CUI INC
20050 SW 112th Ave.
Tualatin, OR 97062
Phone: 503-643-4899
800-275-4899
Fax: 503-372-1266
Website: www.cui.com

TITLE: 2.5mm AUDIO JACK-STEREO		REV: B
PART NO. SJ-2509N	UNITS: MM [INCHES]	
DRAWN BY: ZRJ	APPROVED BY:	SCALE: 3:1

PC FILE NAME: SJ-2509N

FIGURA C.9: Conector de audio Jack 2,5 mm estéreo.



37	69.32	63.50	55.68	55.30	49.68	5.70 +0.25 −0
25	53.03	47.04	39.12	38.84	33.12	5.70 +0.25 −0
15	39.14	33.32	25.18	25.12	19.18	5.90 +0.15 −0
09	30.81	24.99	16.96	16.79	10.96	5.90 +0.15 −0
SHELL	A	B	C	D	E	G

ltr	ecn no	dr	date
D	I06-0074	GVJ	2006/06/08
E	I09-0039	AS	2009/03/16

tolerances unless otherwise specified: angles 0°±1°; linear .00±0.1; .000±.00X

dr	GEORGE.V.JOSEPH	1998/12/16
engr	GEORGE.V.JOSEPH	1998/12/16
chr	ANISH.SALAM	2009/03/16
appd	ANISH.SALAM	2009/03/16

sheet index	revision sheet
	E
	1

FIGURA C.10: Conector DB-9 macho THT.

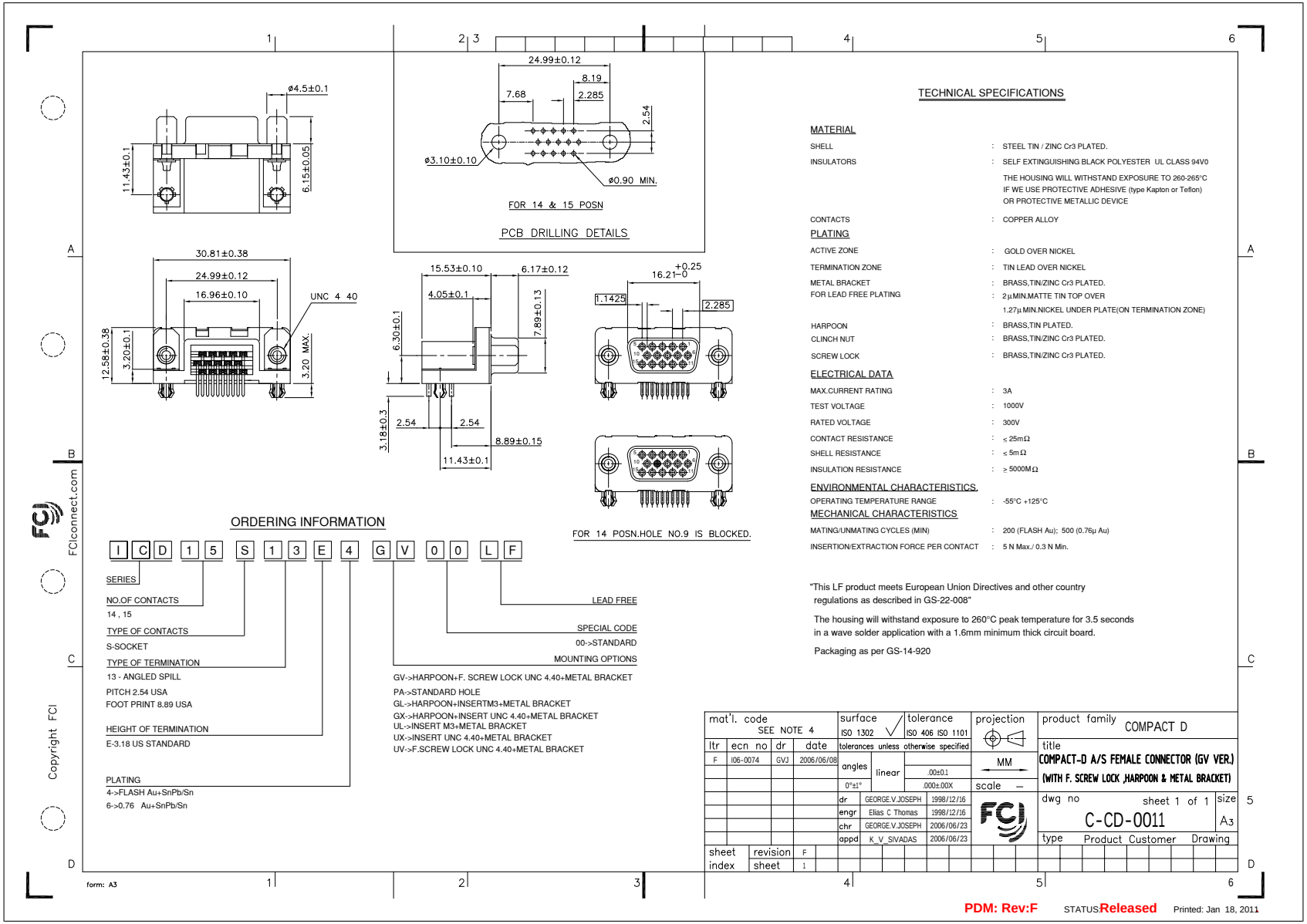


FIGURA C.11: Conector DB-15HD hembra THT.

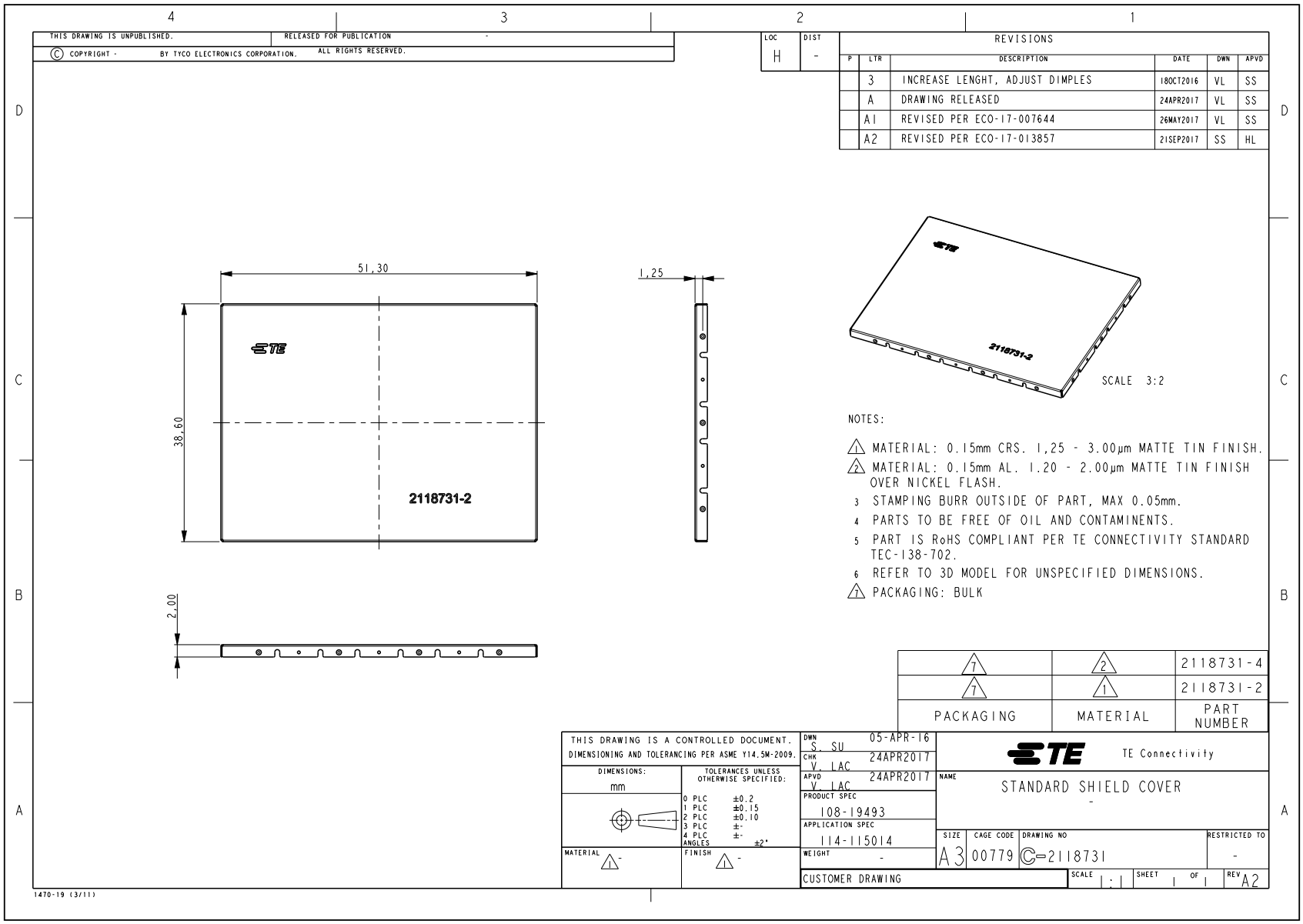


FIGURA C.12: Backlight cover TE_2118731-2.

Apéndice D

Reporte de análisis estático del firmware

Detailed report on module anonymous

Metric	Tag	Overall	Per Function
Lines of Code	LOC	2764	******
McCabe's Cyclomatic Number	MVG	*758*	******
Lines of Comment	COM	785	********
LOC/COM	L_C	3.521	
MVG/COM	M_C	0.966	
Weighted Methods per Class (weighting = unity)	WMC1	146	
Weighted Methods per Class (weighting = visible)	WMCv	113	
Depth of Inheritance Tree	DIT	0	
Number of Children	NOC	0	
Coupling between objects	CBO	0	
Information Flow measure (inclusive)	IF4	0	********
Information Flow measure (visible)	IF4v	0	********
Information Flow measure (concrete)	IF4c	0	********

Definitions and Declarations

Description	LOC	MVG	COM	L_C	M_C
No module extents have been identified for this module					

Functions

Function prototype	LOC	MVG	COM	L_C	M_C
ARRAY_Append(ARRAY *, uint8_t) definition m4/base/array.c:77 declaration m4/base/array.h:49	10	3	0	------	------
ARRAY_AppendBinary(ARRAY *, const uint8_t *, uint32_t) definition m4/base/array.c:107 declaration m4/base/array.h:51	17	8	0	------	******
ARRAY_AppendString(ARRAY *, const char *) definition m4/base/array.c:89 declaration m4/base/array.h:50	16	7	0	------	******
ARRAY_CheckAlnumChars(ARRAY *) definition m4/base/array.c:166 declaration m4/base/array.h:55	*37*	*23*	3	*12.333*	*7.667*
ARRAY_CheckDecimalChars(ARRAY *) definition m4/base/array.c:209	16	7	0	------	******

FIGURA D.1: Análisis estático del firmware con CCCC.

declaration m4/base/array.h:56					
ARRAY_CheckEqualContents(ARRAY *, ARRAY *) definition m4/base/array.c:228 declaration m4/base/array.h:57	9	6	0	------	******
ARRAY_Copy(ARRAY *, ARRAY *) definition m4/base/array.c:239	10	6	0	------	******
ARRAY_Elements(ARRAY *) definition m4/base/array.c:65 declaration m4/base/array.h:47	5	2	0	------	------
ARRAY_Full(ARRAY *) definition m4/base/array.c:71 declaration m4/base/array.h:48	5	2	0	------	------
ARRAY_Init(ARRAY *, uint8_t *, uint32_t) definition m4/base/array.c:36 declaration m4/base/array.h:44	17	6	29	------	0.207
ARRAY_RemoveChars(ARRAY *, uint32_t) definition m4/base/array.c:125 declaration m4/base/array.h:53	23	10	2	*11.500*	5.000
ARRAY_Reset(ARRAY *) definition m4/base/array.c:56 declaration m4/base/array.h:46	8	1	0	------	------
ARRAY_Terminate(ARRAY *) definition m4/base/array.c:154 declaration m4/base/array.h:54	10	3	0	------	------
ARRAY_ToVariant(ARRAY *, VARIANT *) definition m4/base/array.c:252	10	4	0	------	------
BTN_DebouncePressed(uint32_t, uint32_t, uint32_t, uint32_t *) definition m4/base/btn.c:35 declaration m4/base/btn.h:36	19	5	58	------	0.086
BTN_GetState(uint32_t) definition m4/base/btn_lpcopen.c:35 declaration m4/base/btn_io.h:36	5	1	58	------	------
BusyLoop(uint32_t) declaration m0/driver.h:50	1	0	3	------	------
BusyLoop(volatile uint32_t) definition m0/driver.c:256	4	1	7	------	------
COLOR_RGB332_Pal8_Copy(	19	4	0	------	------

Figura D.1: Análisis estático del firmware con CCCC.

COLOR_RGB332_Pal8 *, const COLOR_RGB332_Pal8 *) definition m4/video/color.c:31 declaration m4/video/color.h:33					
COLOR_RGB332_Pal8_GetSel(COLOR_RGB332_Pal8 *, COLOR_RGB332_Pal8_Selector, uint8_t) definition m4/video/color.c:73 declaration m4/video/color.h:38	17	5	0	------	******
COLOR_RGB332_Pal8_Set(...) definition m4/video/color.c:10 declaration m4/video/color.h:31	19	4	0	------	------
COLOR_RGB332_Pal8_Shift(COLOR_RGB332_Pal8 *, int8_t) definition m4/video/color.c:52 declaration m4/video/color.h:35	19	2	0	------	------
CYCLIC_DiscardPending(CYCLIC *) definition m4/base/cyclic.c:213 declaration m4/base/cyclic.h:71	12	3	0	------	------
CYCLIC_In(CYCLIC *, const uint8_t) definition m4/base/cyclic.c:67 declaration m4/base/cyclic.h:56	5	1	0	------	------
CYCLIC_InFromBuffer(CYCLIC *, const uint8_t *, uint32_t) definition m4/base/cyclic.c:73 declaration m4/base/cyclic.h:57	21	7	0	******	******
CYCLIC_InFromStream(CYCLIC *, STREAM_ByteInFunc, void *, uint32_t) definition m4/base/cyclic.c:98 declaration m4/base/cyclic.h:59	28	6	0	******	******
CYCLIC_Init(CYCLIC *, uint8_t *, uint32_t) definition m4/base/cyclic.c:35 declaration m4/base/cyclic.h:53	17	7	30	------	0.233
CYCLIC_Out(CYCLIC *, uint8_t *) definition m4/base/cyclic.c:129 declaration m4/base/cyclic.h:63	5	1	0	------	------
CYCLIC_OutToBuffer(CYCLIC *, uint8_t *, uint32_t) definition m4/base/cyclic.c:137 declaration m4/base/cyclic.h:64	25	9	2	*12.500*	4.500
CYCLIC_OutToStream(CYCLIC *, STREAM_ByteOutFunc, void *, uint32_t) definition m4/base/cyclic.c:167	30	9	0	******	******

Figura D.1: Análisis estático del firmware con CCCC.

declaration m4/base/cyclic.h:66					
CYCLIC_Peek(CYCLIC *, uint32_t) definition m4/base/cyclic.c:201 declaration m4/base/cyclic.h:70	10	3	1	------	------
CYCLIC_Pending(CYCLIC *) definition m4/base/cyclic.c:56 declaration m4/base/cyclic.h:55	9	3	0	------	------
DISPLAY_Clear(uint8_t) definition m4/video/display.c:113 declaration m4/video/display.h:29	9	2	1	------	------
DISPLAY_ClippingRect(uint16_t, uint16_t, uint16_t, uint16_t) definition m4/video/display.c:125 declaration m4/video/display.h:30	24	10	0	******	******
DISPLAY_Get( ) definition m4/video/display.c:69 declaration m4/video/display.h:26	5	1	0	------	------
DISPLAY_Init(DISPLAY *) definition m4/video/display.c:29 declaration m4/video/display.h:25	25	4	5	5.000	------
DISPLAY_MoveClippingToOrigin(int32_t *, int32_t *) definition m4/video/display.c:152 declaration m4/video/display.h:32	16	4	0	------	------
DISPLAY_SetScanlines(uint8_t) definition m4/video/display.c:75 declaration m4/video/display.h:27	9	2	0	------	------
DISPLAY_SwapBuffers( ) definition m4/video/display.c:86 declaration m4/video/display.h:28	18	3	4	------	------
DISPLAY_UndoClippingToOrigin(int32_t *, int32_t *) definition m4/video/display.c:171 declaration m4/video/display.h:33	16	4	0	------	------
DISPLAY_UpdateDriverInfo( ) definition m4/video/display.c:22	5	0	0	------	------
DMA_IRQHandler(void) definition m4/audio/sound.c:249	10	1	0	------	------
DOTMAP_DebugDraw(DOTMAP *, uint32_t, uint32_t)	22	9	2	*11.000*	4.500

Figura D.1: Análisis estático del firmware con CCCC.

definition m4/video/dotmap.c:71 declaration m4/video/dotmap.h:23					
DOTMAP_DebugDrawAround(DOTMAP *, uint32_t, uint32_t) definition m4/video/dotmap.c:100 declaration m4/video/dotmap.h:24	23	7	1	*23.000*	*7.000*
DOTMAP_GetDot(DOTMAP *, int32_t, int32_t) definition m4/video/dotmap.c:25 declaration m4/video/dotmap.h:21	17	7	0	------	******
DOTMAP_Init(DOTMAP *, const uint8_t *, uint32_t, uint32_t, uint32_t) definition m4/video/dotmap.c:7 declaration m4/video/dotmap.h:19	16	7	1	------	*7.000*
DOTMAP_Update(DOTMAP *, int32_t, int32_t) definition m4/video/dotmap.c:46 declaration m4/video/dotmap.h:22	20	2	0	******	------
DisplayLine(uint32_t) declaration m0/driver.h:52	1	0	1	------	------
FONT_DrawGlyph(FONT *, int32_t, int32_t, uint8_t, uint8_t, COLOR_RGB332_Pal8_Selector, uint16_t) definition m4/video/font.c:71 declaration m4/video/font.h:25	*64*	*26*	0	******	******
FONT_DrawText(FONT *, int32_t, int32_t, COLOR_RGB332_Pal8_Selector, int8_t, uint8_t *) definition m4/video/font.c:235 declaration m4/video/font.h:33	*36*	9	0	******	******
FONT_DrawTextLine(FONT *, int32_t, int32_t, COLOR_RGB332_Pal8_Selector, const uint8_t *, uint32_t *) definition m4/video/font.c:140 declaration m4/video/font.h:29	*82*	*25*	4	*20.500*	*6.250*
FONT_Init(FONT *) definition m4/video/font.c:7 declaration m4/video/font.h:22	12	3	0	------	------
FONT_SelectGlyph(FONT *, uint16_t) definition m4/video/font.c:40 declaration m4/video/font.h:24	29	*19*	0	******	******
FONT_SetDefaultGlyphs(FONT *) definition m4/video/font.c:23	15	2	0	------	------

Figura D.1: Análisis estático del firmware con CCCC.

declaration m4/video/font.h:23					
FSM_PutStatusMessage(FEM *, UART *) definition m4/base/fsm_util.c:7 declaration m4/base/fsm_util.h:36	20	0	29	0.690	------
GAMEPAD1_Pressed(...) definition m4/input/gamepads.c:33 declaration m4/input/gamepads.h:29	6	3	0	------	------
GAMEPAD2_Pressed(...) definition m4/input/gamepads.c:40 declaration m4/input/gamepads.h:30	6	3	0	------	------
GAMEPADS_Init(GAMEPADS *) definition m4/input/gamepads.c:6 declaration m4/input/gamepads.h:27	12	3	0	------	------
GAMEPADS_Update(GAMEPADS *) definition m4/input/gamepads.c:21 declaration m4/input/gamepads.h:28	10	2	0	------	------
INDATA_Begin(...) definition m4/base/indata.c:52 declaration m4/base/indata.h:69	13	3	0	------	------
INDATA_Init(INDATA *, UART *) definition m4/base/indata.c:37 declaration m4/base/indata.h:68	12	4	29	------	------
LINE_Draw(int16_t, int16_t, int16_t, int16_t, COLOR_RGB332_Pal8_Selector) definition m4/video/line.c:13 declaration m4/video/line.h:7	*80*	*27*	0	******	******
M0APP_IRQHandler( ) definition m4/video/display.c:13	5	0	2	------	------
RIT_IRQHandler( ) definition m0/driver.c:47 definition m0/driver.c:262	25	4	12	2.083	------
SDCARD_Init(SDCARD *) definition m4/storage/sdcard.c:43 declaration m4/storage/sdcard.h:27	13	4	0	------	------
SDCARD_Update( ) definition m4/storage/sdcard.c:59 declaration m4/storage/sdcard.h:28	9	2	0	------	------
SOUND_Init(SOUND *) definition m4/audio/sound.c:261	*36*	4	1	**36.000**	------

Figura D.1: Análisis estático del firmware con CCCC.

declaration m4/audio/sound.h:107					
SOUND_Mute(bool) definition m4/audio/sound.c:306 declaration m4/audio/sound.h:108	14	7	1	------	*7.000*
SOUND_PauseBGM( ) definition m4/audio/sound.c:392 declaration m4/audio/sound.h:112	14	5	0	------	******
SOUND_PlayBGM(const char *, uint32_t) definition m4/audio/sound.c:341 declaration m4/audio/sound.h:110	24	5	0	******	******
SOUND_PlaySFX(const SOUND_SFX *, bool) definition m4/audio/sound.c:426 declaration m4/audio/sound.h:114	*52*	*17*	4	*13.000*	4.250
SOUND_ResumeBGM( ) definition m4/audio/sound.c:409 declaration m4/audio/sound.h:113	14	5	0	------	******
SOUND_StopBGM( ) definition m4/audio/sound.c:369 declaration m4/audio/sound.h:111	19	6	0	------	******
SOUND_Update( ) definition m4/audio/sound.c:324 declaration m4/audio/sound.h:109	14	3	0	------	------
SYSTICK_GetTickRateMicroseconds( ) definition m4/base/systick.c:66 declaration m4/base/systick.h:40	5	1	0	------	------
SYSTICK_Now( ) definition m4/base/systick.c:72 declaration m4/base/systick.h:41	5	1	0	------	------
SYSTICK_SetHook(SYSTICK_HookFunc) definition m4/base/systick.c:78 declaration m4/base/systick.h:42	7	1	0	------	------
SYSTICK_SetMicrosecondPeriod(uint32_t) definition m4/base/systick.c:51 declaration m4/base/systick.h:38	6	0	0	------	------
SYSTICK_SetMillisecondPeriod(uint32_t) definition m4/base/systick.c:59 declaration m4/base/systick.h:39	6	0	0	------	------
SysTick_Handler( ) definition m4/base/systick.c:40	8	1	0	------	------

Figura D.1: Análisis estático del firmware con CCCC.

TILEMAP_Draw(const TILEMAP *, int32_t, int32_t, uint16_t, uint16_t, TILEMAP_TileProc) definition m4/video/tilemap.c:16 declaration m4/video/tilemap.h:52	*80*	*23*	6	*13.333*	3.833
TILEMAP_DrawRepeat(...) definition m4/video/tilemap.c:116 declaration m4/video/tilemap.h:55	*66*	*18*	5	*13.200*	3.600
TILE_Draw(const uint8_t *, int32_t, int32_t) definition m4/video/tile.c:5 declaration m4/video/tile.h:7	**116**	**35**	1	**116.000**	**35.000**
UART_Config(UART *, uint32_t) definition m4/base/uart_lpcopen.c:35 declaration m4/base/uart_io.h:38	15	4	60	------	------
UART_GetByte(void *) definition m4/base/uart_lpcopen.c:55 declaration m4/base/uart_io.h:40	14	5	1	------	5.000
UART_Init(UART *, void *, uint32_t) definition m4/base/uart.c:36 declaration m4/base/uart.h:62	14	3	29	------	------
UART_PutBinary(UART *, const uint8_t *, uint32_t) definition m4/base/uart.c:75 declaration m4/base/uart.h:66	11	3	0	------	------
UART_PutByte(void *, uint8_t) definition m4/base/uart_lpcopen.c:71 declaration m4/base/uart_io.h:41	14	4	0	------	------
UART_PutMessage(UART *, const char *) definition m4/base/uart.c:87 declaration m4/base/uart.h:68	5	1	0	------	------
UART_PutMessageArgs(...) definition m4/base/uart_util.c:44 declaration m4/base/uart_util.h:36	*44*	*14*	72	0.611	0.194
UART_PutStatusMessage(UART *) definition m4/base/uart_util.c:96 declaration m4/base/uart_util.h:38	25	2	0	******	------
UART_Recv(UART *) definition m4/base/uart.c:105 declaration m4/base/uart.h:70	10	3	0	------	------
UART_RecvDiscardPending(UART *) definition m4/base/uart.c:139 declaration m4/base/uart.h:73	9	3	0	------	------

Figura D.1: Análisis estático del firmware con CCCC.

UART_RecvInjectByte(UART *, uint8_t) definition m4/base/uart.c:117 declaration m4/base/uart.h:71	9	3	0	------	------
UART_RecvPeek(UART *, uint32_t) definition m4/base/uart.c:128 declaration m4/base/uart.h:72	9	3	0	------	------
UART_RecvPendingCount(UART *) definition m4/base/uart.c:64 declaration m4/base/uart.h:65	9	3	0	------	------
UART_Send(UART *) definition m4/base/uart.c:93 declaration m4/base/uart.h:69	10	3	0	------	------
UART_SendPendingCount(UART *) definition m4/base/uart.c:53 declaration m4/base/uart.h:64	9	3	0	------	------
USBMS_Init(USBMS *) definition m4/storage/usbms.c:49 declaration m4/storage/usbms.h:27	13	4	0	------	------
USBMS_Update( ) definition m4/storage/usbms.c:65 declaration m4/storage/usbms.h:28	9	2	0	------	------
VARIANT_CmpStrings(VARIANT *, VARIANT *) definition m4/base/variant.c:209	25	*15*	0	******	******
VARIANT_CmpUint32s(VARIANT *, VARIANT *) definition m4/base/variant.c:240	8	4	0	------	------
VARIANT_SetFloat(VARIANT *, float) definition m4/base/variant.c:57	8	1	0	------	------
VARIANT_SetInt32(VARIANT *, int32_t) definition m4/base/variant.c:47	8	1	0	------	------
VARIANT_SetPointer(VARIANT *, void *) definition m4/base/variant.c:67	8	1	0	------	------
VARIANT_SetString(VARIANT *, const char *) definition m4/base/variant.c:77	8	1	0	------	------
VARIANT_SetUint32(VARIANT *, uint32_t) definition m4/base/variant.c:37	8	1	29	------	------
VARIANT_ToFloat(VARIANT *) definition m4/base/variant.c:143	21	9	0	******	******

Figura D.1: Análisis estático del firmware con CCCC.

VARIANT_ToInt32(VARIANT *) definition m4/base/variant.c:115	21	9	0	******	******
VARIANT_ToString(VARIANT *) definition m4/base/variant.c:171	29	9	0	******	******
VARIANT_ToUint32(VARIANT *) definition m4/base/variant.c:87	21	9	0	******	******
a2d(char) definition m4/base/printf.c:97	10	10	0	------	******
a2i(char, char **, int, int *) definition m4/base/printf.c:108	14	4	0	------	------
defaultTileProc(const uint8_t **, uint16_t, int32_t, int32_t, uint32_t, uint32_t) definition m4/video/tilemap.c:6	8	1	0	------	------
fillRxBuffer( ) definition m4/audio/sound.c:14	*59*	10	23	2.565	0.435
i2a(int, char *) definition m4/base/printf.c:88	8	1	0	------	------
initHardware( ) definition m0/driver.c:15 definition m0/driver.c:221	*35*	0	8	4.375	------
init_printf(...) definition m4/base/printf.c:215 declaration m4/base/printf.h:111	6	0	102	------	------
li2a(long, char *) definition m4/base/printf.c:59	8	1	0	------	------
main( ) definition m0/driver.c:74 definition m0/driver.c:293	**148**	*22*	33	4.485	0.667
mixSFX( ) definition m4/audio/sound.c:185	*35*	9	1	**35.000**	*9.000*
mixSFX_16BitFullRate(SOUND_SFX_Slot *) definition m4/audio/sound.c:166	13	2	3	------	------
mixSFX_16BitHalfRate(SOUND_SFX_Slot *) definition m4/audio/sound.c:146	14	2	5	------	------
mixSFX_16BitQuadRate(SOUND_SFX_Slot *) definition m4/audio/sound.c:124	16	2	7	------	------

Figura D.1: Análisis estático del firmware con CCCC.

mixSFX_8BitHalfRate(SOUND_SFX_Slot *) definition m4/audio/sound.c:103	15	2	5	------	------
putchw(void *, putcf, int, char, char *) definition m4/base/printf.c:123	12	5	0	------	******
putcp(void *, char) definition m4/base/printf.c:229	4	0	0	------	------
schedulerAddTask(...) definition m4/base/copos.c:179 declaration m4/base/copos.h:32	25	5	41	0.610	0.122
schedulerDeleteTask(uint32_t) definition m4/base/copos.c:234 declaration m4/base/copos.h:37	20	4	16	1.250	------
schedulerDispatchTasks(uint32_t) definition m4/base/copos.c:117 declaration m4/base/copos.h:31	19	3	9	------	------
schedulerInit(void) definition m4/base/copos.c:59 declaration m4/base/copos.h:28	9	1	7	------	------
schedulerModifyTaskPeriod(uint32_t, uint32_t) definition m4/base/copos.c:210 declaration m4/base/copos.h:36	14	5	0	------	******
schedulerReportStatus(void) definition m4/base/copos.c:272 declaration m4/base/copos.h:38	21	4	15	1.400	------
schedulerStart(uint32_t) definition m4/base/copos.c:105 declaration m4/base/copos.h:29	6	0	6	------	------
schedulerUpdate(uint32_t) definition m4/base/copos.c:77 declaration m4/base/copos.h:30	17	4	8	------	------
startDMAtransfer( ) definition m4/audio/sound.c:228	8	0	0	------	------
statusUpdateCallback(...) definition m4/storage/sdcard.c:9 definition m4/storage/usbms.c:9	*58*	*14*	2	*29.000*	*7.000*
swapBuffers( ) definition m4/audio/sound.c:238	9	0	0	------	------
swap_i16(int16_t *, int16_t *)	6	0	0	------	------

Figura D.1: Análisis estático del firmware con CCCC.

definition m4/video/line.c:5					
tfp_format(...) declaration m4/base/printf.h:116	1	0	0	------	------
tfp_format(void *, putcf, char *, va_list) definition m4/base/printf.c:136	*64*	*16*	0	******	******
tfp_printf(...) definition m4/base/printf.c:221 declaration m4/base/printf.h:113	8	0	0	------	------
tfp_sprintf(...) definition m4/base/printf.c:236 declaration m4/base/printf.h:114	9	0	0	------	------
ui2a(int, int, int, char *) definition m4/base/printf.c:70	17	7	0	------	******
uli2a(int, int, int, char *) definition m4/base/printf.c:41	17	7	0	------	******
waitForHsyncSignal( ) definition m0/driver.c:282	8	1	1	------	------

Relationships

Clients	Suppliers

Figura D.1: Análisis estático del firmware con CCCC.

Apéndice E

Temporizados alternativos para señal de video

E.1. Blanking reducido (CVT-R)

El temporizado CVT-R reduce considerablemente los tiempos de blanking de un modo de video volviéndolo incompatible con los monitores CRT. Sin embargo, en los monitores LCD compatibles la reducción de estos tiempos es beneficiosa ya que se reduce la frecuencia de *pixel clock* y por ende el ancho de banda de la señal.

Para obtener los parámetros de un modo de video compatible con *reduced blanking* se llama a la aplicación cvt pasando como parámetro la resolución activa, la frecuencia de refresco y la opción *–reduced* como se muestra en el listado E.1.

LISTADO E.1: Uso del comando CVT para generar temporizado con reduced blanking.

```
1 usuario@linux:$ cvt --verbose --reduced 1280 720 60
2 # 1280x720 59.74 Hz (CVT 0.92M9-R) hsync: 44.27 kHz; pclk: 63.75 MHz
3 Modeline "1280x720R"   63.75  1280 1328 1360 1440  720 723 728 741 +hsync -vsync
```

La aplicación devuelve el temporizado en formato *Modeline* para incluir directamente en el archivo de configuración de xorg-server. En la tabla E.1 se analizan los parámetros del *Modeline* según su significado.

TABLA E.1: Significado de los parámetros del *Modeline* generado por CVT con *reduced blanking* activado.

Valor	Nombre (xorg)	Significado
Modeline		Inicia definición de modo de video en xorg.conf
"1280x720R"	mode name	Nombre de la definición
63.75	dot clock	Pixel clock, frecuencia de pixel, 63,75 MHz
1280	hdisp	Pixeles activos en horizontal
1328	hsyncstart	En que pixel comienza señal de sincro. horizontal
1360	hsyncend	En que pixel termina señal de sincro. horizontal
1440	htotal	Total de pixeles en horizontal (activos + blanking)
720	vdisp	Pixeles activos en vertical
723	vsyncstart	En que linea comienza señal de sincro. vertical
728	vsyncend	En que linea termina señal de sincro. vertical
741	vtotal	Total de lineas verticales (activas + blanking)
+hsync		Señal de sincro. horizontal con polaridad positiva
-vsync		Señal de sincro. vertical con polaridad negativa

El modo de video resultante tiene un total de 1440x741 pixeles contando los intervalos de Blanking. Cada cuadro de video se actualiza a una frecuencia de $59{,}74\,\mathrm{Hz}$ y equivale a la frecuencia de sincronismo vertical ($\overline{VSYNC}$). La frecuencia de sincronismo horizontal (*HSYNC*) es de $44{,}27\,\mathrm{Hz}$ y la de pixel (*PCLK*) es $63{,}75\,\mathrm{MHz}$.

Las ecuaciones en E.1 (*PCLK*), E.2 (*HSYNC*) y E.3 ($\overline{VSYNC}$) corroboran estos datos e ilustran sobre la relación existente entre los parámetros.

$$PCLK = 1440 \cdot 741 \cdot 59{,}74\,\mathrm{Hz} = 63\,744\,969{,}6\,\mathrm{Hz} = 63{,}75\,\mathrm{MHz} \tag{E.1}$$

$$HSYNC = \frac{63\,744\,969{,}6\,\mathrm{Hz}}{1440} = 44\,267{,}34\,\mathrm{Hz} = 44{,}27\,\mathrm{kHz} \tag{E.2}$$

$$\overline{VSYNC} = \frac{63\,744\,969{,}6\,\mathrm{Hz}}{1440 \cdot 741} = 59{,}74\,\mathrm{Hz} \tag{E.3}$$

En la ecuación E.4 (*TPIXEL*) se calcula el periodo de un solo pixel.

$$TPIXEL = \frac{1}{63\,744\,969{,}6\,\mathrm{Hz}} = 1{,}568\,751\,238\,37 \cdot 10^{-8}\,\mathrm{s} = 15{,}69\,\mathrm{ns} \tag{E.4}$$

Utilizando el valor de (*TPIXEL*), en la tabla E.2 se calcula la duración del Front Porch, sincronismos, Back Porch y demás tiempos que serán útiles en la implementación del driver del adaptador de video.

TABLA E.2: Intervalos de tiempo en la señal de video con reduced blanking. Números truncados al quinto decimal.

Parámetro	**Intervalo**	**Cantidad**	**Tiempo**
Horizontal Total (Line)	1440_{htotal}	1440 pixeles	22,590 01 µs
Horizontal Active Pixels	1280_{hdisp}	1280 pixeles	20,080 01 µs
Horizontal Blanking Interval	$1440_{\mathrm{htotal}} - 1280_{\mathrm{hdisp}}$	160 pixeles	2,510 00 µs
Horizontal Front Porch	$1328_{\mathrm{hsyncstart}} - 1280_{\mathrm{hdisp}}$	48 pixeles	0,753 00 µs
Horizontal Sync Width	$1360_{\mathrm{hsyncend}} - 1328_{\mathrm{hsyncstart}}$	32 pixeles	0,502 00 µs
Horizontal Back Porch	$1440_{\mathrm{htotal}} - 1360_{\mathrm{hsyncend}}$	80 pixeles	1,255 00 µs
Vertical Active Lines	720_{vdisp}	720 lineas	16,264 81 ms
Vertical Blanking Interval	$741_{\mathrm{vtotal}} - 720_{\mathrm{vdisp}}$	21 lineas	0,474 39 ms
Vertical Front Porch	$723_{\mathrm{vsyncstart}} - 720_{\mathrm{vdisp}}$	3 lineas	0,067 77 ms
Vertical Sync Width	$728_{\mathrm{vsyncend}} - 723_{\mathrm{vsyncstart}}$	5 lineas	0,112 95 ms
Vertical Back Porch	$741_{\mathrm{vtotal}} - 728_{\mathrm{vsyncend}}$	13 lineas	0,293 67 ms

Un pixel lógico equivale a 5x5 pixeles iguales en la señal de video. Se lo divide verticalmente en 5 partes de este modo: 5x1, 5x2, 5x3, 5x4 y 5x5. Cada una de esas partes se dibuja en una linea de 720p o scanline y demora el tiempo especificado en la ecuación E.5.

$$\begin{aligned} TLP5 &= TPIXEL \cdot 5 \\ &= 1{,}568\,751\,238\,37 \cdot 10^{-8}\,\mathrm{s} \cdot 5 \\ &= 7{,}843\,756\,191\,85 \cdot 10^{-8}\,\mathrm{s} = 78{,}437\,561\,918\,5\,\mathrm{ns} \end{aligned} \tag{E.5}$$

El dibujo de pixeles se implementa en assembler. Si bien no es completamente determinista, a los fines prácticos se asumirá como cierto que una instrucción de n ciclos de reloj demora $n \cdot MCU_{cycle}$. La constante de duración de un ciclo de reloj fue definida en la ecuación 3.2 como $MCU_{cycle} = 4{,}901\,960\,784\,31\,\mathrm{ns}$.

En la ecuación E.6 se calcula la cantidad de ciclos de reloj por cada 5 pixeles del modo de video.

$$\begin{aligned} TLP5_{cycles} &= \frac{TLP5}{MCU_{cycle}} \\ &= \frac{78{,}437\,561\,918\,5\,\mathrm{ns}}{4{,}901\,960\,784\,31\,\mathrm{ns}} \\ &= 16{,}001\,262\,631\,4 \end{aligned} \tag{E.6}$$

A diferencia del temporizado CVT calculado en el Capitulo 3 e implementado en el Capitulo 4, en este modo alternativo de blanking reducido $TLP5_{cycles}$ es un valor entero de ciclos de reloj del microcontrolador.

Bibliografía

[1] Alec Bourque. *Página web del proyecto Uzebox*. URL: http://belogic.com/uzebox.

[2] Alec Bourque. *Uzebox - How It Works*. Ver. 10. The Uzebox Project. 2010. URL: http://uzebox.org/files/wiki/uzebox_how_it_works_v10.pdf.

[3] Foley, van Dam, Feiner y Hughes. *Computer Graphics, Principles and practice - Second edition in C*. Addison Wesley, 1997.

[4] Santiago Germino. *Informe de Avance del Trabajo Final de la Carrera de Especialización en Sistemas Embebidos. Consola de videojuegos basada en EDU-CIAA (RETRO-CIAA)*. Ver. 1.1. Ago. de 2018. URL: http://www.retro-ciaa.com/docs/retrociaa_informe_avance_08_2018.pdf.

[5] Santiago Germino. *Informe de Avance del Trabajo Final de la Carrera de Especialización en Sistemas Embebidos. Consola de videojuegos basada en EDU-CIAA (RETRO-CIAA)*. Ver. 1.2. Oct. de 2018. URL: http://www.retro-ciaa.com/docs/retrociaa_informe_avance_10_2018.pdf.

[6] Santiago Germino. *Plan de Proyecto del Trabajo Final de la Carrera de Especialización en Sistemas Embebidos. Consola de videojuegos basada en EDU-CIAA (RETRO-CIAA)*. Ver. 1.8. Mayo de 2018. URL: http://www.retro-ciaa.com/docs/retrociaa_plan_de_proyecto.pdf.

[7] Ing. Hernan Felipe Rey Hernandez. *MEMORIA DEL TRABAJO FINAL. Desarrollo de interfaz gráfica para EDU-CIAA*. Dic. de 2016. URL: http://laboratorios.fi.uba.ar/lse/tesis/LSE-FIUBA-Trabajo-Final-CESE-Felipe-Rey-2016.pdf.

[8] Ing. Hernan Felipe Rey Hernandez. *Video de prueba de dispositivo DisplayLink conectado a EDU-CIAA-NXP*. URL: https://youtu.be/kmSHw2GcvZE.

[9] Josh Hintze y André LaMothe. *Inside the XGS PIC 16-Bit - User manual and programming guide*. Ver. 1.0. Nurve Networks LLC. 2009. URL: http://www.ic0nstrux.com/download/xgs_pic/xgs_pic_user_manual_sample.pdf.

[10] Espressif Systems Inc. *ESP32 AT Instruction Set and Examples*. Ver. 1.1. 2018. URL: https://www.espressif.com/sites/default/files/documentation/ESP32_AT_Instruction_Set_and_Examples__EN.pdf.

[11] Espressif Systems Inc. *ESP32 Hardware Design Guidelines*. Ver. 2.5. 2018. URL: https://www.espressif.com/sites/default/files/documentation/esp32_hardware_design_guidelines_en.pdf.

[12] Texas Instruments Incorporated. *PCM5100A Datasheet*. Ver. C. 2016.

[13] André LaMothe. *Game programming for the propeller powered HYDRA*. Ver. 1.0.1. Nurve Networks LLC. 2006. URL: https://www.parallax.com/sites/default/files/downloads/32360-Hydra-Game-Dev-Manual-v1.0.1.pdf.

[14] André LaMothe. *Página web de HYDRA Game Development Kit*. URL: http://www.ic0nstrux.com/hydra-game-development-kit.

[15] André LaMothe. *Página web de XGameStation PIC 16-Bit*. URL: http://www.ic0nstrux.com/products/gaming-systems/xgs-pic-16-bit-development-system.

[16] HDMI Licensing LLC. *High-Definition Multimedia Interface Specification*. Ver. 1.0. 9-2003. URL: https://glenwing.github.io/docs/HDMI-1.0.pdf.

[17] Telmo Moya. *Repositorio de codigo - Speecy AMP y Speecy ILI9341*. URL: https://github.com/telmomoya/educiaa-bm.

[18] Telmo Moya. *Video de Speecy ILI9341 (LCD)*. URL: https://youtu.be/v_a6dzCKRAI.

[19] Robert L. Myers. *Display Interfaces - Fundamentals and standards*. Wiley-SID, 2002.

[20] STMicroelectronics N.V. *1-Mbit serial I2C bus EEPROM Datasheet*. Ver. 14. 2017.

[21] Eric Pernia, Martin Ribelotta y contribuidores. *Repositorio GitHub de la futura version 3 del firmware del proyecto CIAA*. URL: https://github.com/epernia/cese-edu-ciaa-template.

[22] NXP Semiconductors. *LPC435x/3x/2x/1x 32-bit ARM Cortex-M4/M0 MCU*. Ver. 5.3.

[23] NXP Semiconductors. *LPC43xx/LPC43Sxx ARM Cortex-M4/M0 multi-core microcontroller*. Ver. 2.3.

[24] Stockman, MacLeod y Johnson. *2-deg cone fundamentals (based on the Stiles and Burch 2-deg CMFs)*. 1993. URL: http://www.cvrl.org/database/text/cones/smj2.htm.

[25] VESA. *VESA Enhanced Display Data Channel (EDDC) Standard*. Ver. 1.2. URL: https://glenwing.github.io/docs/VESA-EDDC-1.2.pdf.

[26] Joseph Yiu. *The Definitive Guide to ARM Cortex-M0 and Cortex-M0+ Processors - 2nd Ed.* Elsevier Inc, 2015.

[27] Joseph Yiu. *The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors - 3rd Ed.* Elsevier Inc, 2014.