Escola Universitària
Politècnica de Mataró

Centre adscrit a:

UNIVERSITAT POLITÈCNICA
DE CATALUNYA

GRAU EN ENGINYERIA INFORMÀTICA

INTERNET DE LES COSES. MICROCONTROLADORES Y *SINGLE-BOARD COMPUTERS*

AUTOR: JONATHAN ALCÁZAR MORA
PONENT: DR. LÉONARD JANER

PRIMAVERA 2015

TecnoCampus
Mataró-Maresme

Dedicatoria

Dedico este proyecto a mi familia, sin su esfuerzo y apoyo no estaría hoy aquí.

Agradecimientos

A mi tutor, Dr Leonard Janer, por guiarme y aconsejarme a lo largo de todo el proyecto.

Resumen

Este proyecto trata sobre los microcontroladores y *single-board computers*. La finalidad del mismo es dar a conocer dichos dispositivos, sus posibles apliciones y ventajas/desventajas frente a otros equipos, así como hacer pruebas y analizar el rendimiento de alguno de los dispositivos analizados.

Resum

Aquest projecte tracta sobre els microcontroladors i *single-board computers*. La finalidad del mateix és donar a coneixer aquests dispositius, les seves possibles aplicacions i les avantatges/desavantatges vers altres equips, així com realitzar proves i analitzar el rendiment d'alguns dels dispositius analitzats.

Abstract

This project is about the microcontrollers and single-board computers, the objective of which is to announce those devices, their pros and cons from similar computers, their possible applications as well as develop some tests to see their performance.

Contenido

A continuación citamos la finalidad de cada capítulo.

Introducción

- Introducir y explicar qué es una arquitectura de computador.
- Describir la arquitectura RISC. Qué es y cómo funciona.
- Describir la arquitectura CISC. Qué es y cómo funciona.
- Comparar las arquitecturas RISC y CISC.
- Describir la arquitectura ARM. Qué es y cómo funciona.

Microcontroladores

- Introducir y explicar qué es un microcontrolador.
- Describir las características de un microcontrolador. Para qué sirven y cómo son.
- Explicar las diferencias entre un microcontrolador y un *single-board computer*.

- Realizar un análisis de los datos técnicos de un microcontrolador. Características del procesador, memoria, almacenamiento...

Single-board computers

- Introducir y explicar qué es un *single-board computer*.
- Describir las características de un *single-board computer*. Para qué sirven y cómo son.
- Enunciar las utilidades que le podemos dar a un single-board computer

Clasificación de dispositivos

- Enunciar y explicar los criterios de clasificación que se han tenido en cuenta a la hora de seleccionar los dispositivos con los que vamos a tratar.

Dispositivos analizados

- Realizar un análisis técnico de cada dispositivo seleccionado.

Casos prácticos

- Desarrollar tres casos prácticos con los dispositivos seleccionados.

Índice general

Índice general I

Índice de figuras VII

Índice de cuadros XI

Índice de código XIII

1. Objetivos 1

1.1. Propósito 1

1.2. Finalidad 1

1.3. Objeto 1

1.4. Alcance 2

2. Introducción 3

2.1. ¿Qué es una arquitectura? 3

2.2. Arquitectura RISC 4

2.3. Arquitectura CISC 4

2.4. Ejemplo práctico 5

2.5. RISC vs CISC 6

2.6. Tabla comparativa RISC vs CISC 7

2.7. Arquitectura ARM 7

3. Microcontroladores **9**

3.1. Introducción 9

3.2. ¿Qué es un microcontrolador? 10

3.3. Características de un microcontrolador 10

3.4. Diferencia entre un microcontrolador y un *single-board computer* . . 11

3.5. Datos técnicos de un microcontrolador 12

4. *Single-board computers* **15**

4.1. Introducción 15

4.2. ¿Qué es un *single-board computer*? 15

4.3. Características de un *single-board computer* 16

4.4. Datos técnicos de un *single-board computer* 17

4.5. Utilidades de un *single-board computer* 17

5. Clasificación de dispositivos **19**

5.1. Introducción 19

5.2. Criterios de clasificación 19

5.2.1. Tipo de dispositivo 19

5.2.2. Orientación 20

5.2.3. *Open/Closed hardware* 20

5.2.4. *Open/Closed source* 20

5.2.5. Especificaciones técnicas 20

6. Dispositivos analizados **23**

6.1. Introducción 23

6.2. Arduino 24

6.2.1. Arquitectura 24

6.2.2. Precio 24

6.2.3. *Inputs/Outputs* 25

6.2.4. Puertos 25

6.2.5. Procesador y memoria 26

6.2.6. Almacenamiento 27

6.2.7. Periféricos y módulos 27

6.2.8. Consumo y alimentación 27

6.2.9. Sistema operativo 27

6.3. Raspberry Pi 28
6.3.1. Arquitectura 28
6.3.2. Precio 28
6.3.3. *Inputs/Outputs* 29
6.3.4. Puertos 29
6.3.5. Procesador y memoria 30
6.3.6. Almacenamiento 30
6.3.7. Periféricos y módulos 31
6.3.8. Consumo y alimentación 31
6.3.9. Sistema operativo 31
6.4. Odroid 33
6.4.1. Arquitectura 33
6.4.2. Precio 33
6.4.3. *Inputs/Outputs* 34
6.4.4. Puertos 34
6.4.5. Procesador y memoria 35
6.4.6. Almacenamiento 35
6.4.7. Periféricos y módulos 36
6.4.8. Consumo y alimentación 36

6.4.9. Sistema operativo 36

7. Casos prácticos **37**

7.1. Servidor de archivos 37

7.1.1. Introducción 37

7.1.2. Odroid 38

7.1.3. Raspberry pi 42

7.1.4. Análisis 46

7.1.5. Conclusión 48

7.2. Servidor web 48

7.2.1. Introducción 48

7.2.2. Odroid 49

7.2.3. Raspberry pi 49

7.2.4. Análisis 49

7.2.5. Conclusión 54

7.3. Media center 55

7.3.1. Introducción 55

7.3.2. Odroid 55

7.3.3. Raspberry pi 55

7.3.4. Análisis 56

7.3.5. Conclusión 63

8. Conclusiones **65**

Bibliografía **67**

Índice de figuras

2.1. Circuito integrado de un microprocesador 3

2.2. Procesador ARM (RISC) en unaimpresora HP 4

2.3. Familias ARM 8

3.1. Procesador de un microcontrolador genérico 9

3.2. Esquema de un computador 10

3.3. Microcontrolador 11

3.4. *Single-board computer* 12

4.1. *Single-board computer* 15

6.1. Placa Arduino Uno 24

6.2. Esquema de puertos. Arduino Duemilanove 26

6.3. Placa Raspberry pi 2 b B 28

6.4. Esquema de puertos. Raspberry Pi 2 30

6.5. Placa Odroid C1 33

6.6. Esquema de puertos. Odroid C1 35

7.1. Odroid. Conectar a unidad de red 40

7.2. Odroid. Configuración unidad de red 41

7.3. Odroid. Contraseña servidor 41

7.4. Odroid. Carpeta compartida 42

7.5. Raspberry. Conectar a unidad de red 45

7.6. Raspberry. Configuración unidad de red 45

7.7. Raspberry. Contraseña servidor 46

7.8. Raspberry. Carpeta compartida 46

7.9. Raspberry. Prueba de velocidad 47

7.10. Odroid. Prueba de velocidad 47

7.11. Odroid. Gráfica peticiones/ms 51

7.12. Raspberry. Gráfica peticiones/ms 52

7.13. apache.org. Gráfica peticiones/ms 53

7.14. Kodi. Rendimiento reproductor 57

7.15. Kodi. Menú 58

7.16. Kodi. Pictures 59

7.17. Kodi. Videos 60

7.18. Kodi. Reproductor 60

7.19. Kodi. Video add-ons 61

7.20. Kodi. Youtube Video 62

7.21. Kodi. Music add-ons 62

7.22. Kodi. Music streaming 63

Índice de cuadros

2.1. Cuadro comparativo RISC vs CISC 7

Índice de código

7.1. Odroid. fdisk -l 38
7.2. Odroid. /etc/fstab 39
7.3. Odroid. mkdir and chown 39
7.4. Odroid. mount 39
7.5. Odroid. install samba 39
7.6. Odroid. /etc/samba/smb.conf 40
7.7. Odroid. smbpasswd 40
7.8. Odroid. samba restart 40
7.9. Raspberry. fdisk -l 42
7.10. Raspberry. /etc/fstab 43
7.11. Raspberry. mkdir and chown 43
7.12. Raspberry. mount 43
7.13. Raspberry. install samba 43
7.14. Raspberry. /etc/samba/smb.conf 44
7.15. Raspberry. smbpasswd 44
7.16. Raspberry. samba restart 44
7.17. Raspberry. install apache2 49
7.18. Odroid. install apache2 49
7.19. ab command 49

1. Objetivos

1.1. Propósito

El propósito de este proyecto es el de hacer una introducción a los *single-board computers* y a los microcontroladres, así como explicar su utilidad y realizar una serie de pruebas para analizar el comportamiento y rendimiento de los mismos.

1.2. Finalidad

Dar a conocer tanto los *single-board computers* como los microcontroladores así como en qué situaciones nos pueden ser de utilidad.

1.3. Objeto

Conocer la existencia de estos dispositivos, para qué sirven y sus limitaciones.

1.4. Alcance

El proyecto estará dividido en dos partes. La primera será una parte puramente teórica y la segunda será una parte práctica así como de análisis.

2. Introducción

Para entender mejor el funcionamiento de los microcontroladores y computadores, debemos primero analizar la arquitectura de sus procesadores. No obstante, antes de entrar en terminología RISC y CISC debemos comprender a qué nos referimos al hablar de la arquitectura de un procesador.

2.1. ¿Qué es una arquitectura?

Un procesador, en su interior, no es más que un conjunto de bloques interconectados entre si. Cada uno de estos realiza una función. El diseño de esos elementos y como se interconectan es lo que se llama arquitectura.

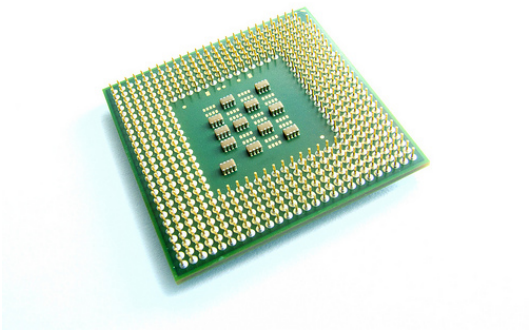

Figura 2.1: Circuito integrado de un microprocesador

Para funcionar, una computadora lee instrucciones y datos. La velocidad a la que lee datos y realiza cálculos, viene determinada por la potencia del microprocesador.

En otras palabras, un microprocesador CISC, al igual que uno RISC, estará diseñado

tanto a nivel de hardware como de software para que la ejecución de ciertas instrucciones en lenguaje máquina se haga de manera óptima.

2.2. Arquitectura RISC

La principal característica que identifica a la arquitectura RISC *(Reduced Instruction Set Computer)* es que utiliza instrucciones pequeñas y simples que duran siempre lo mismo, un ciclo de reloj. Para ello tiene incorporado en el microprocesador una gran cantidad de memoria con la finalidad de agilizar los accesos a las instrucciones y variables, consiguiendo así más velocidad de procesamiento.

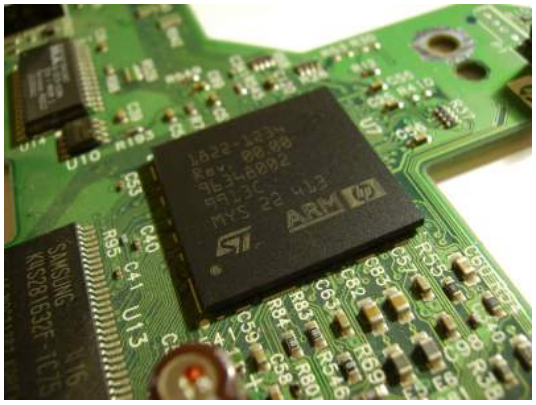

Figura 2.2: Procesador ARM (RISC) en unaimpresora HP

Esta arquitectura está orientada a ejecutar instrucciones simples de manera rápida y eficiente, característica que la hace idónea para los 3.Microcontroladores, donde la tarea a realizar está muy acotada.

2.3. Arquitectura CISC

La arquitectura CISC *(Complex Instruction Set Computer)* no dispone de tanta memoria dentro del microprocesador, por tanto los tiempos de lectura/escritura son mayores, así como el tiempo de ejecución de las instrucciones. A cambio, nos da la posibilidad de ejecutar programas complejos más fácil y eficientemente que la arquitectura RISC, debido a que dispone de más instrucciones.

Se utiliza en la mayoría de 4.*Single-board computers*, puesto que necesitamos una arquitectura que nos permita desarrollar una gran variedad de tareas.

2.4. Ejemplo práctico

A fin de entender correctamente cómo funciona una arquitectura RISC y una CISC vamos a utilizar un ejemplo muy simple. El de preparar una tortilla.

En términos CISC tendríamos menos instrucciones pero a su vez más complejas.

- Paso 1. Batir los huevos.
- Paso 2. Preparar la sartén.
- Paso 3. Verter los huevos en la sartén.
- Paso 4. Esperar unos minutos.
- Paso 5. Retirar la tortilla.

Mientras que en arquitectura RISC tendríamos un mayor núimero de instrucciones pero más simples.

- Paso 1. Sacar los huevos de la nevera.
- Paso 2. Romper los huevos y verterlos en un plato.
- Paso 3. Batirlos hasta que salga espuma.
- Paso 4. Echar aceite en una sartén.
- Paso 5. Calentar el aceite.
- Paso 6. Verter los huevos en la sartén.

- Paso 7. Esperar unos minutos.
- Paso 8. Darle la vuelta a la tortilla.
- Paso 9. Esperar unos minutos.
- Paso 10. Retirar la tortilla

2.5. RISC vs CISC

Los términos reducido (RISC) y complejo (CISC) explican la característica más definitiva para cada arquitectura.

Sin embargo, hoy en día, para dos procesadores de las mismas características, un procesador RISC tiene una capacidad de procesamiento de dos a cuatro veces mayor que la de un CISC. Además, su estructura hardware es tan simple que ocupa una fracción de la superfície ocupada por el circuito integrado de un procesador CISC.

Esto nos lleva a pensar que tarde o temprano la arquitectura RISC sustituirá completamente a la arquitectura CISC, pero esto no será realmente así. La tendencia futura será dejar de crear núcleos totalmente CISC para dar paso a núcleos híbridos RISC/-CISC.

2.6. Tabla comparativa RISC vs CISC

RISC	**CISC**
Instrucciones simples de un ciclo de reloj	Instrucciones complejas de más de un ciclo de reloj
Sólo las intrucciones LOAD/STORE referencian a la memoria	Cualquier instrucción puede referenciar a la memoria
Instrucciones ejecutadas por el hardware	Instrucciones interpretadas por el microprograma[1]
Instrucciones de formato fijo	Instrucciones de formato variable
Pocas instrucciones y modos	Muchas instrucciones y modos
La complejidad se encuentra en el compilador	La complejidad se encuentra en el microprograma
Varios grupos de registros	Un solo grupo de registros
Ejecución de las instrucciones concurrente	Ejecución de las instrucciones poco concurrente

Cuadro 2.1: Cuadro comparativo RISC vs CISC

2.7. Arquitectura ARM

Una vez conocemos las arquitecturas RISC y CISC es imprescindible hablar de la arquitectura más utilizada en los dispositivos que vamos a utilizar en este proyecto. ARM (*Advanced Risc Machine*) es una arquitectura RISC de 32 bits y recientemente de 64 bits desarollada por la empresa *ARM Holdings*. La relativa simplicidad de estos procesadores los hace ideales para aplicaciones de baja potencia.[7]

[1]Microprograma: Set de instrucciones en lenguaje máquina que definen las operaciones que un computador tiene que llevar a cabo.

ARM tiene sus orígenes en la pasada década de los 80, momento en el que la compañía Acorn Computers decidió emprender un proyecto para desarrollar un procesador de gran potencia para sus equipos informáticos. El primer procesador ARM, ARM2, vio la luz comercialmente en 1986.[4]

Los procesadores ARM se distribuyen por familias, y cada familia tiene unas propiedades distintas(bits, caché, MHz, MIPS...). En la figura 2.3.Familias ARM vemos algunas de las familias (ARMv4T, ARMv5TJ, ARMv6...) junto con sus procesadores (ARM926, Cortex-A9, Cortex-M3...)

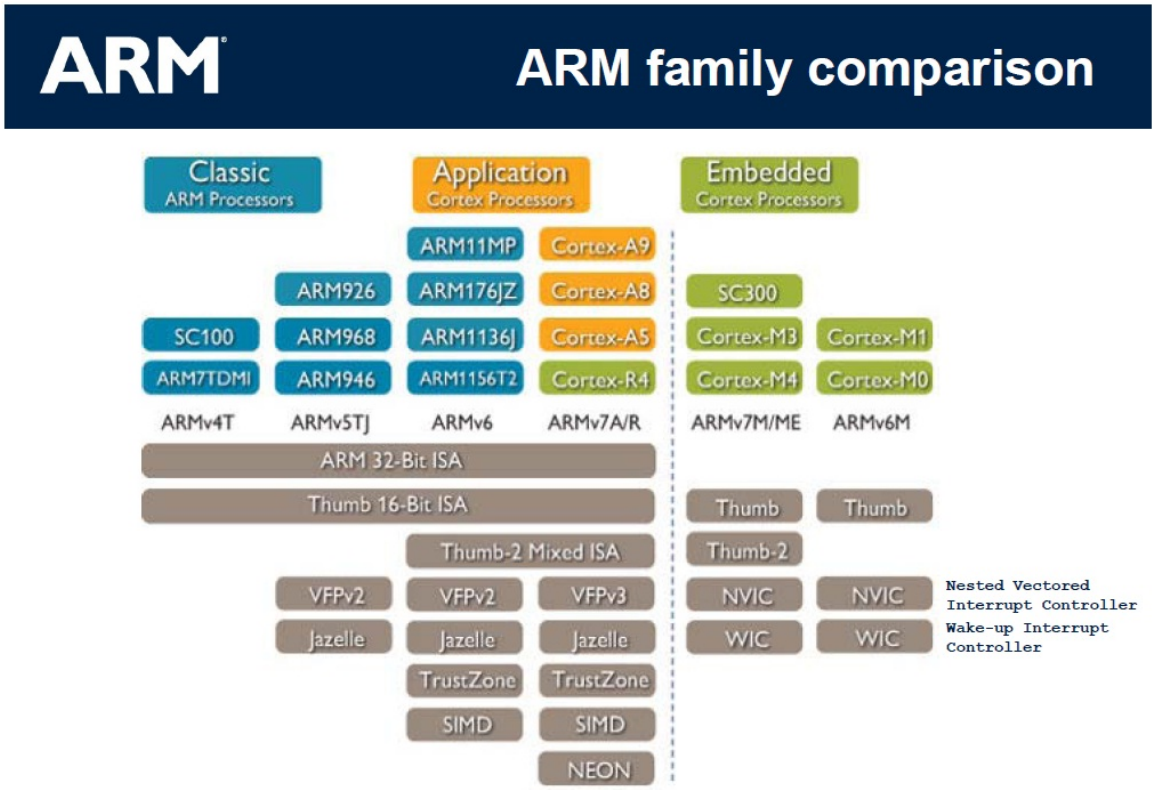

Figura 2.3: Familias ARM

Actualmente la arquitectura ARM está presente en el mundo de los smartphones, tablets, smart watchs, videoconsolas portátiles, calculadoras y un sin fin de dispositivos más.

3. Microcontroladores

3.1. Introducción

Hoy en día los microcontroladores están presentes en una gran parte de los dispositivos que nos rodean en la vida cotidiana. El mando de la televisión que utilizamos cada día, por ejemplo, no es más que un microcontrolador con un sensor de infrarrojos y botones como entrada de información. De igual manera, todos los coches modernos tienen integrados también un gran número de microcontroladores, los cuales se encargan desde el control del motor hasta el sistema antibloqueo de los frenos, el control de crucero, etc.

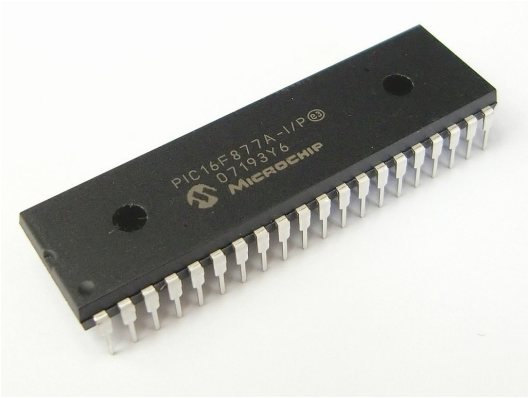

Figura 3.1: Procesador de un microcontrolador genérico

Y podríamos seguir también con un sin fin de dispositivos como los teléfonos móviles, lavadoras, neveras, lavavajillas...

3.2. ¿Qué es un microcontrolador?

Un microcontrolador es un ordenador, y todos los ordenadores, ya hablemos de un ordenador de sobremesa, un portátil, un smart phone... Todos tienen las siguientes características en común.

- Poseen una unidad de procesamiento central[10]. Esta unidad es la encargada de interpretar los programas informáticos así como ejecutar las instrucciones de los mismos.
- Poseen una memoria de acceso aleatorio[9]. Utilizada para cargar las instrucciones y variables de los programas que posteriormente se van a ejecutar.
- Poseen dispositivos de entrada/salida[8], que permiten la comunicación ya sea con seres vivos o con otros dispositivos.

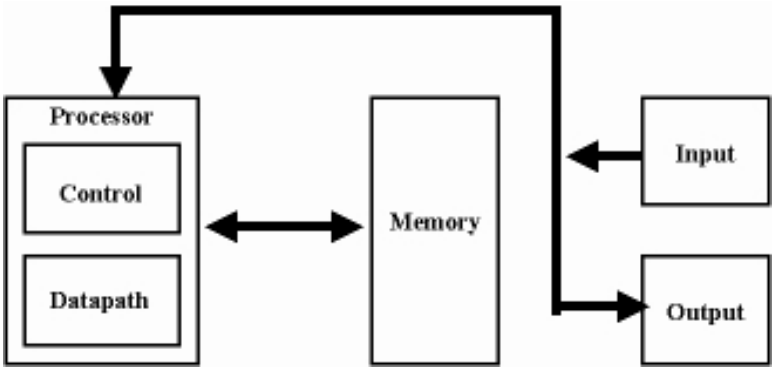

Figura 3.2: Esquema de un computador

3.3. Características de un microcontrolador

Las características o atributos principales de un microcontrolador son las siguientes.

- Se integran en los productos con la finalidad de controlar propiedades o acciones de dicho producto. Sería un ejemplo válido el control de velocidad de crucero de un automóvil.
- Están destinados a realizar una o varias tareas. Un microcontrolador programado para controlar la pantalla de un reloj digital solo servirá para eso.
- La mayoría tienen un consumo de energía muy bajo debido a la poca potencia de procesamiento y capacidad de sus componentes.
- La mayoría son de dimensiones reducidas y de bajo coste, importante a la hora de facilitar la implementación en otros dispositivos.
- En algunos casos tienen que trabajar en condiciones adversas. Sería el caso de los microcontroladores que actúan sobre los motores de los automóviles, teniendo que soportar altas temperaturas.

3.4. Diferencia entre un microcontrolador y un *single-board computer*

La diferencia es que un ordenador portátil o de sobremesa es un dispositivo genérico que no está especializado en ninguna tarea en concreto. Esto quiere decir que puede servir para reproducir multimedia, jugar a videojuegos, navegar por internet...

En cambio, un microcontrolador tiene la tarea de controlar las acciones o propiedades de otro dispositivo. Además,

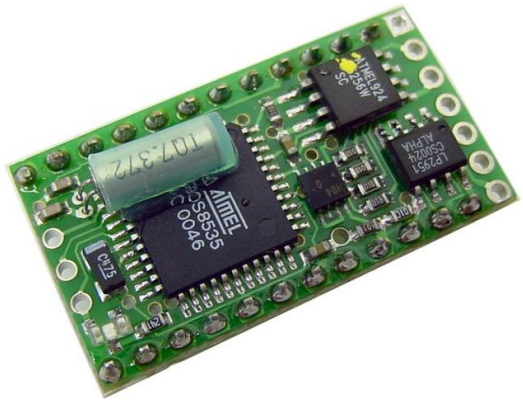

Figura 3.3: Microcontrolador

si lo programamos para realizar una o varias tareas solo podrá usarse para eso.

Si utilizamos el ejemplo de un microondas con una pantalla LED o LCD, no tendría sentido integrar un ordenador genérico para controlar dicha pantalla, ya que además de aumentar el tamaño y el coste del producto final se desaprovecharía el 99 % de la potencia del ordenador.

Por contra, si implementamos un microcontrolador con las especificaciones justas y necesarias (CPU, memoria y entradas/salidas) tendremos un computador de tamaño reducido, económico y enfocado sólo a realizar la tarea en la que estamos interesados.

Figura 3.4: *Single-board computer*

3.5. Datos técnicos de un microcontrolador

La variedad de procesadores que se pueden implementar en un microcontrolador es muy amplia. Dicha variedad puede ir de simples procesadores de 4bits a procesadores complejos de 64bits. No obstante, lo más común son procesadores simples y baratos.

En cuanto a la memoria, un microcontrolador de categoría media-baja suele tener

1000bytes de memoria ROM, 20bytes de memoria RAM y 8 pines de entrada/salida. Un microcontrolador así puede llegar a costar céntimos, de esta manera no encarece el producto final.

4. *Single-board computers*

4.1. Introducción

Como hemos expuesto anteriormente todo computador tiene las siguientes partes: procesador, memoria y entradas/salidas. No es una excepción el tipo de dispositivo en el que nos vamos a centrar. Los *single-board computers*.

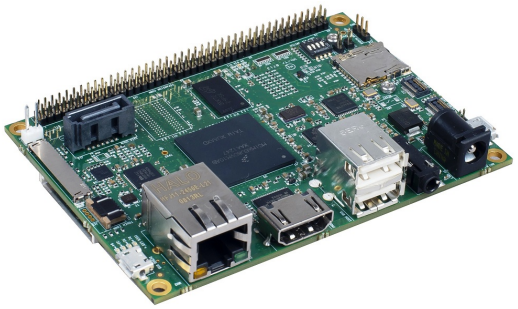

Figura 4.1: *Single-board computer*

4.2. ¿Qué es un *single-board computer*?

Un *single-board computer* es un ordenador totalmente funcional que tiene todos sus componentes implementados en una única placa.

Estos equipos no cuentan con ningún tipo de ranura de expansión y no se les puede sustituir ningún componente, no obstante la mayoría de modelos cuentan con módulos o periféricos que se pueden comunicar con la placa para cubrir alguna necesidad

concreta.

Principalmente fueron inventados con fines educacionales, de desarrollo o para controlar otros dispositivos. Sin embargo, cada vez son más potentes y se les están dando usos más variados.

4.3. Características de un *single-board computer*

Las principales características de un single board-computer son las siguientes.

- Tienen un tamaño muy reducido, entorno a 90mm de alto por 60mm de ancho.
- El precio de la mayoría de dispositivos ronda los 50$.
- El consumo es muy bajo. Tomando por ejemplo una fuente de alimentación de un ordenador de sobremesa standard de 500W, una Raspberry Pi consume 3W. Es decir, 167 veces menos.
- Disponen de periféricos y módulos para ampliar sus utilidades.
- No son dispositivos modulares, es decir, no podemos sustituir la tarjeta gráfica, procesador, memoria... por una de diferentes características en caso sea necesario.
- Sus especificaciones son inferiores a las de un ordenador de sobremesa o portátil.
- Apenas se calientan, por tanto, pueden prescindir de disipadores minimizando así el ruido.
- Disponen de entradas/salidas para controlar otros dispositivos.

4.4. Datos técnicos de un *single-board computer*

La variedad de single-board computers disponibles en el mercado es muy grande, dependiendo de la gama de la placa los precios y características pueden variar bastante. Teniendo en cuenta esto, en la actualidad la mayoría de dispositivos cumplen con un gran número de las características citadas a continuación. 1GB de memoria RAM.

- De 2 a 4 Cores con una frecuencia de 1GHz aproximadamente.
- De 20 a 40 GPIOs.
- Alimentación de 5V.
- De 2 a 4 USB's 2.0.
- Ethernet a 100Mb o 1000Mb
- Salida de vídeo HDMI
- Ranura para tarjeta de memoria SD/MicroSD/eMMC

4.5. Utilidades de un *single-board computer*

Asumiendo que tenemos un ordenador que nos cabe en la palma de la mano las aplicaciones que podemos encontrar son muchas, sin embargo, las más conocidas son las siguientes.

- Mini ordenador de sobremesa conectando un monitor.
- Servidor web para sitios pequeños y con pocas funcionalidades.
- Aplicaciones domóticas.

- Aplicaciones de reconocimiento de voz e imagen.
- Servidor de archivos.
- Media center.
- Servidor multimedia.

5. Clasificación de dispositivos

5.1. Introducción

Una vez comprendemos qué es un microcontrolador y qué es un computador vamos a clasificar algunos de los dispositivos más populares y utilizados teniendo en cuenta sus características y posibles usos.

5.2. Criterios de clasificación

A continuación explicaremos los criterios que se han tenido en cuenta para clasificar los dispositivos contemplados en esta memoria.

5.2.1. Tipo de dispositivo

Este primer gran grupo hace referencia a si estamos tratando con un microcontrolador o con un computador.

5.2.2. Orientación

Este criterio indica a qué tipo de usuario está orientado el dispositivo. Normalmente va ligado a la facilidad de configuración y uso

5.2.3. *Open/Closed hardware*

Determina si el fabricante permite que terceros desarrollen hardware compatible con el dispositivo.

5.2.4. *Open/Closed source*

De igual manera que con el hardware, si el fabricante distribuye y facilita el código fuente de su producto para que terceros desarrollen software asumiremos que se trata de una placa open source, de lo contrario, estaremos hablando de una placa closed source.

5.2.5. Especificaciones técnicas

Otra manera de hacer una clasificación viene dada por las especificaciones técnicas del dispositivo y si estas se adaptan a nuestras necesidades. A continuación veremos las que hemos considerado más importantes.

- **Arquitectura:** Arquitectura del procesador. Importante a la hora de ejecutar ciertas instrucciones o aplicaciones.
- **Precio:** Precio del dispositivo. Directamente proporcional a las especificaciones del mismo.

- ***Inputs/Outputs:*** Entradas y salidas que tiene. Dependerá de si queremos controlar más o menos dispositivos.
- **Procesador:** Velocidad del procesador. Si tenemos la necesidad de hacer muchos cálculos u operaciones nos decantaremos a un dispositivo con más procesadores y más potentes
- **Memoria:** Cantidad y velocidad de la misma. Importante a la hora de almacenar mucha información en la memoria temporal.
- **Puertos:** Cantidad y tipo de puertos que dispone el dispositivo. USB, Ethernet...
- **Almacenamiento:** Capacidad de almacenamiento de datos y cómo.
- **Periféricos/Módulos:** Periféricos o módulos.
- **Consumo:** Consumo eléctrico. Dato a tener en cuenta cuando, por ejemplo, necesitamos 10 dispositivos iguales encendidos las 24 horas.
- **Tamaño:** Dimensiones del dispositivo. En caso de que lo queramos implementar dentro de otro dispositivo
- **Alimentación:** Alimentación requerida por la placa.
- **Sistema operativo:** Sistemas operativos soportados. La mayoría son sistemas UNIX[1]

[1] Ver enlace: `http://www.unix.org/what_is_unix.html`

6. Dispositivos analizados

6.1. Introducción

Existe una gran variedad de dispositivos en el mercado, no obstante, hemos elegido los que nos han parecido más atractivos para analizarlos y realizar una serie de casos prácticos para medir su rendimiento y utilidad.

Nos centraremos básicamente en tres dispositivos:

- **Arduino:** Por ser el microcontrolador más implementado y conocido.
- **Raspberry Pi:** Por ser el *single-board computer* más famoso en su clase.
- **Odroid**: Por que, pese a no ser altamente conocido, ofrece mejores características por el mismo precio que sus competidores.

6.2. Arduino

El microcontrolador por excelencia, Arduino es una plataforma open source, orientada al desarrollo y basada en una placa con un microcontrolador y herramientas para programar el mismo.

Los microcontroladores que implementa Arduino pertenecen en su gran mayoría a la familia Atmel[1] y dependiendo el modelo encontraremos más o menos entradas/salidas en la placa.

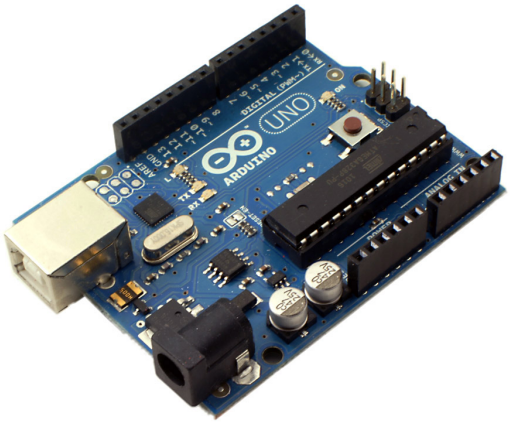

Figura 6.1: Placa Arduino Uno

6.2.1. Arquitectura

Los microcontroladores Atmel AVR son una familia de microcontroladores RISC. Ésta arquitectura se caracteriza por sus instrucciones pequeñas, simples y rápidas de ejecutar.

6.2.2. Precio

El rango de precios de las placas Arduino va de los 12€ a los 60€. Donde por 20€ podemos encontrar la placa más estandard con un microcontrolador Atmega 328, 26 entradas/salidas diferentes, entrada de alimentación y USB para programar.

[1] Ver enlace: http://www.atmel.com/products/microcontrollers/avr/

6.2.3. *Inputs/Outputs*

Al tratarse de un microcontrolador, los inputs y outputs se convierten en una de las partes esenciales. Dentro de las placas Arduino podemos encontrar tres tipos diferentes de entrada/salida.

- **Digital:** Una señal digital se utiliza para expresar dos estados, ON (1) y OFF (0). Un ejemplo de salida digital sería el estado de un LED, encendido (1) o apagado (1).
- **Analógica:** A diferencia de la salida/entrada digital, la analógica expresa un valor, por ejemplo, la temperatura captada por un sensor.
- **PWM:** PWM o Pulse Width Modulation es una técnica para transferir información mediante una señal digital cuadrada. Una posible aplicación de dicha técnica sería regular la intensidad con al que brilla un LED. [1]

En cuanto al número de entradas/salidas implementado de cada tipo varia según el modelo.

- **Digitales:** De 9 a 54
- **Analógicas:** De 4 a 16
- **PWM:** De 5 a 14.

6.2.4. Puertos

Además de las entradas/salidas de las que hemos hablado en el capítulo anterior 6.2.3.*Inputs/Outputs* Arduino cuenta con los siguientes puertos:

- **Puerto USB:** Se utiliza para conectar la placa a un ordenador y programar su comportamiento.
- **Puerto de alimentación:** Puerto para alimentar el dispositivo.

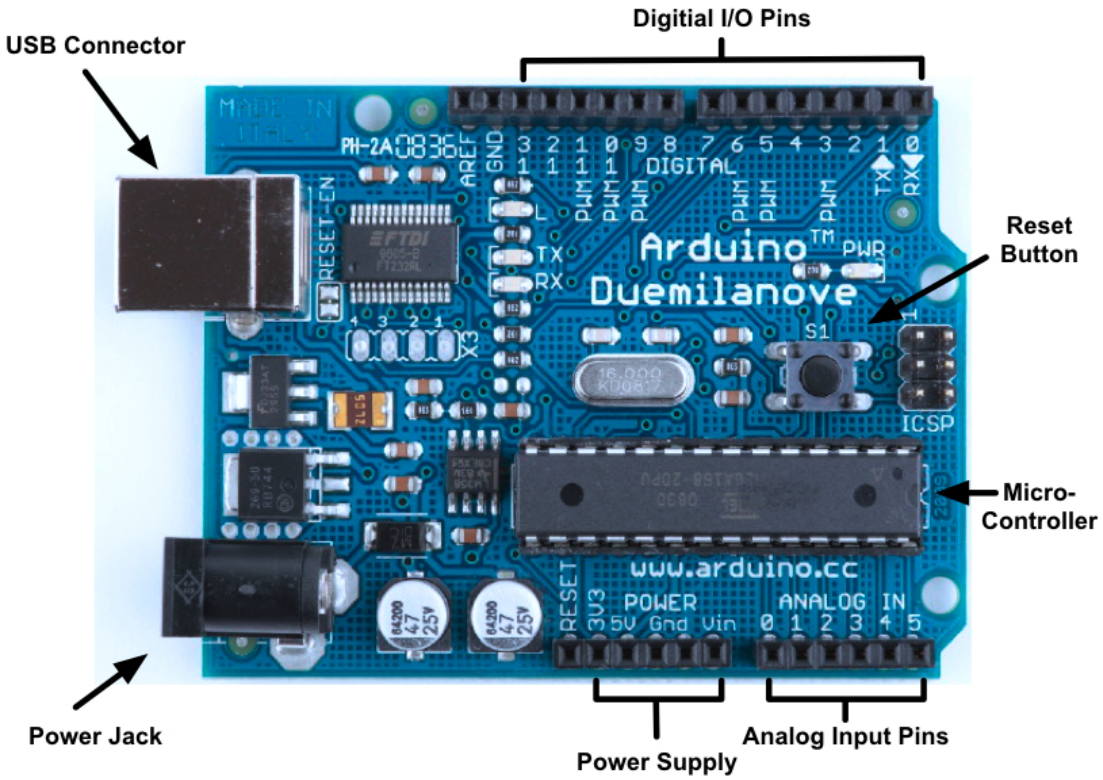

Figura 6.2: Esquema de puertos. Arduino Duemilanove

6.2.5. Procesador y memoria

La velocidad del procesador de una Arduino está entre 8MHz y 84MHz. La memoria EEPROM está en 1KB y la memoria SRAM puede variar de 1-96KB.

Como podemos observar la potencia de procesamiento de un microcontrolador es muy baja, pero como hemos explicado en el capítulo 3.Microcontroladores no vamos a utilizar un dispositivo de estas características para ejecutar un navegador de internet, un sistema operativo completo (Windows, Linux...) o un editor de texto.

6.2.6. Almacenamiento

Gran parte del almacenamiento de las placas Arduino está dedicado a los programas e información que necesitan los mismos. Arduino consta de una memoria flash que va de 16KB a 512KB.

6.2.7. Periféricos y módulos

Cuenta con distintos módulos compatibles con sus placas. La mayoría se utilizan para ampliar la conectividad, como sería el caso del módulo Ethernet, WiFi, USB, etc.

6.2.8. Consumo y alimentación

Dependiendo de la potencia de la placa podemos encontrar voltajes variados. El rango estaría entre 2,7V-5,5V mientras que la alimentación vendría dada por un jack conectado directamente a la placa que suministre de 5,5V a 12V.

6.2.9. Sistema operativo

Arduino cuenta con sistemas operativos muy simples que pueden llegar a gestionar archivos, ejecutar programas y jugar a videojuegos, sin embargo, el punto fuerte de Arduino sigue siendo controlar otros dispositivos.

6.3. Raspberry Pi

Raspberry Pi, el más famoso en su categoría, es un *single-board computer* de bajo coste que conectado a un monitor puede desarrollar las mismas tareas que desarrolla un ordenador de sobremesa. Está orientado al aprendizaje y es un proyecto *open software*.

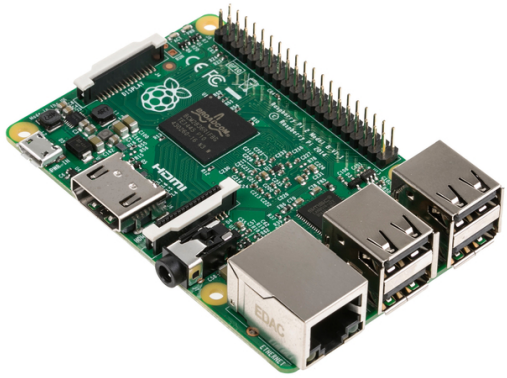

Figura 6.3: Placa Raspberry pi 2 b B

Raspberry Pi dispone de varios modelos, de más antiguo a más actual: Modelo A+[2], Modelo B+[3], y modelo 2B[4]. Nosotros nos centraremos en el modelo más actual, el 2B.

6.3.1. Arquitectura

El procesador del dispositivo está basado en la arquitectura ARM que a su vez forma parte de la familia RISC.

6.3.2. Precio

El precio de la placa sin ningún accesorio es de 35$.

[2]Ver enlace: https://www.raspberrypi.org/products/model-a-plus/
[3]Ver enlace: https://www.raspberrypi.org/products/model-b-plus/
[4]Ver enlace: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

6.3.3. *Inputs/Outputs*

Aunque este dispositivo no está orientado completamente a controlar dispositivos, como sería el caso de los microcontroladores, dispone también de entradas/salidas. En concreto 40 de propósito general (GPIOs)[2]. Este tipo de entrada/salida trabaja con señales digitales, sin embargo, mediante un conversor se puede llegar transformar la señal digital en analógica.

6.3.4. Puertos

Además de las entradas/salidas de las que hemos hablado en el capítulo anterior 6.3.3.*Inputs/Outputs* Raspberry Pi tiene los siguientes puertos:

- **Puerto de alimentación:** Puerto para alimentar el dispositivo.
- **Puerto ethernet:** Conexión ethernet a 100Mb/s.
- **Puertos USB:** Un total de 4 puertos USB 2.0.
- **Puerto HDMI:** Puerto full HDMI para conectar el dispositivo a un monitor.
- **Puerto *DSI Display Connector:*** Permite conectar una pantalla LCD mediante un cable plano de 15 pines.
- **Puerto *CSI Connector Camera:*** Permite conectar una cámara.
- **Puerto de audio y video:** Salida alternativa al puerto HDMI. Es un jack de 3.5 mm.

Un dato a destacar sobre los puertos de este dispositivo es que comparte el ancho de banda para los puertos USB y Ethernet, en términos prácticos esto significa que si quiero compartir un disco duro externo USB a través de la red, el ancho de banda de lectura/escritura del disco duro y de la red estarán compartidos.

Esta característica hace que la Raspberry Pi no sea un buen dispositivo para aplicaciones que hagan un uso intensivo del puerto ethernet, como sería el caso de un servidor de archivos.

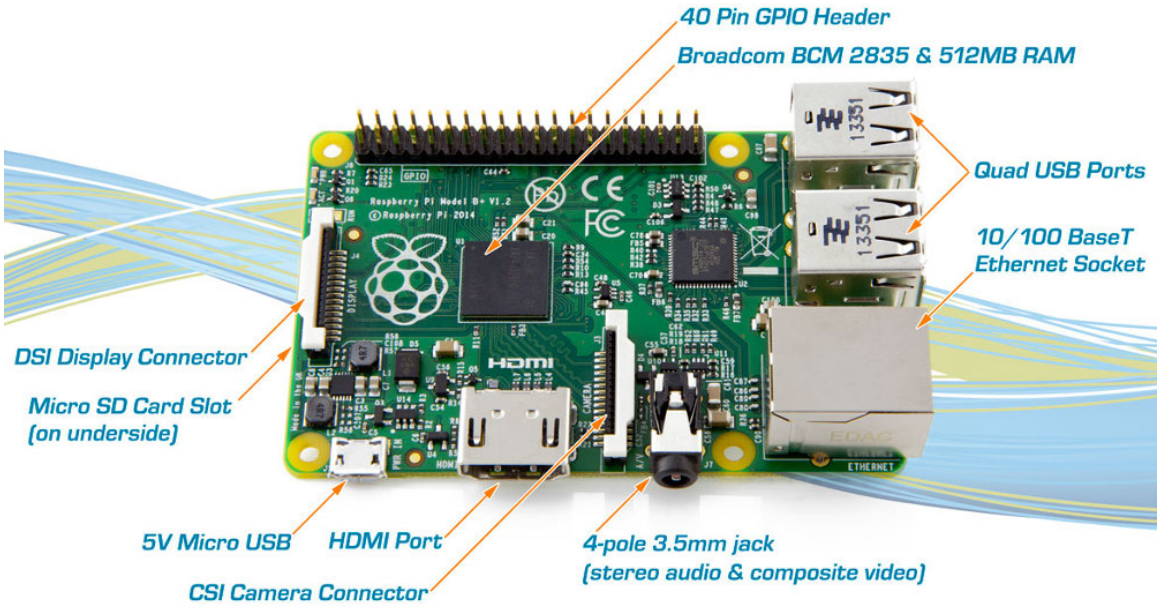

Figura 6.4: Esquema de puertos. Raspberry Pi 2

6.3.5. Procesador y memoria

La Raspberry Pi cuenta con un Cortex-A7 quad-core a 900MHz que, comparado con su predecesor de un único core a 700MHz supone un gran avance a la hora de ejecutar programas, reproducir multimedia, etc. En cuanto a la memoria cuenta con 1GB a una velocidad de 400 MHz.

6.3.6. Almacenamiento

Tanto el sistema operativo como los datos que queramos almacenar en la placa estarán en la tarjeta micro SD, no obstante, mediante los puertos USB que implementa la placa podremos conectar dispositivos de almacenamiento externos como *pen drives*

y discos duros.

6.3.7. Periféricos y módulos

Teniendo en cuenta que estamos corriendo un sistema operativo normal y corriente podemos conectar la gran mayoría de dispositivos como si de un ordenador normal se tratase. Desde keyboards, mouses, unidades de almacenamiento, tarjetas Wi-Fi, altavoces, auriculares a pantallas y todo tipo de sensores.

6.3.8. Consumo y alimentación

La Raspberry Pi está alimentada por conexión micro USB de 5V y tiene un consumo medio de 800mA (4W).

6.3.9. Sistema operativo

Al utilizar una arquitectura ARM puede ejecutar todas los distribuciones GNU/Linux. Podemos considerar que la más conocidas son:

- Raspbian[5]
- Ubuntu[6]
- OpenELEC[7]
- OSMC[8]

[5]Ver enlace: `http://www.raspbian.org/`
[6]Ver enlace: `https://www.raspberrypi.org/downloads/`
[7]Ver enlace: `http://openelec.tv/`
[8]Ver enlace: `https://osmc.tv/`

- RiscOS[9]
- Pidora[10]

[9] Ver enlace: `https://www.riscosopen.org/content/`
[10] Ver enlace: `http://pidora.ca/`

6.4. Odroid

De la compañía HardKernel[11], aunque no es tan sonada como su competidor directo Raspberry Pi, se dedican a la creación de *single-board computers* de características superiores y precio similar a sus competidores. Las odroid son placas de desarrollo y open hardware/software.

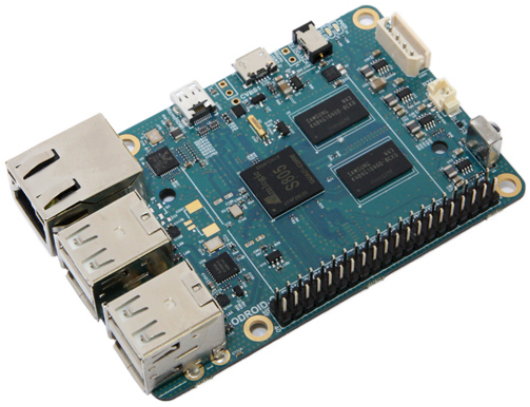

Figura 6.5: Placa Odroid C1

Los modelos más conocidos son: Odroid XU3, Odroid U3 y Odroid C1. Nosotros nos vamos a centrar en la Odroid C1.

6.4.1. Arquitectura

ARMv7. Ver sección: 6.3.1.Arquitectura.

6.4.2. Precio

El precio de la placa sin accesorios es de 35$.

[11] Ver enlace: `http://www.hardkernel.com/`

6.4.3. *Inputs/Outputs*

Al igual que su competidor, dispone de 40 GPIO's[2] que trabajan con señales digitales.

6.4.4. Puertos

Además de las entradas/salidas de las que hemos hablado en el capítulo anterior 6.4.3.*Inputs/Outputs* Odroid tiene los siguientes puertos:

- **Puerto de alimentación:** Mediante un connector de 0.8mm/2.5mm de diametro y 5V/2A
- **Puerto ethernet:** Conexión ethernet a 1000Mb/s
- **Puerto USB:** USB 2.0. Un total 4.
- **Puerto Micro HDMI:** Para conectar un monitor.
- **Puerto Micro USB OTG:** Útil para conectar periféricos.
- **Puerto para batería RTC:** Necesario para conectar una batería RTC y mantener el dispositivo en hora.
- **Puerto *serial console*:** Conectándolo a un ordenador te da acceso a una consola linux.

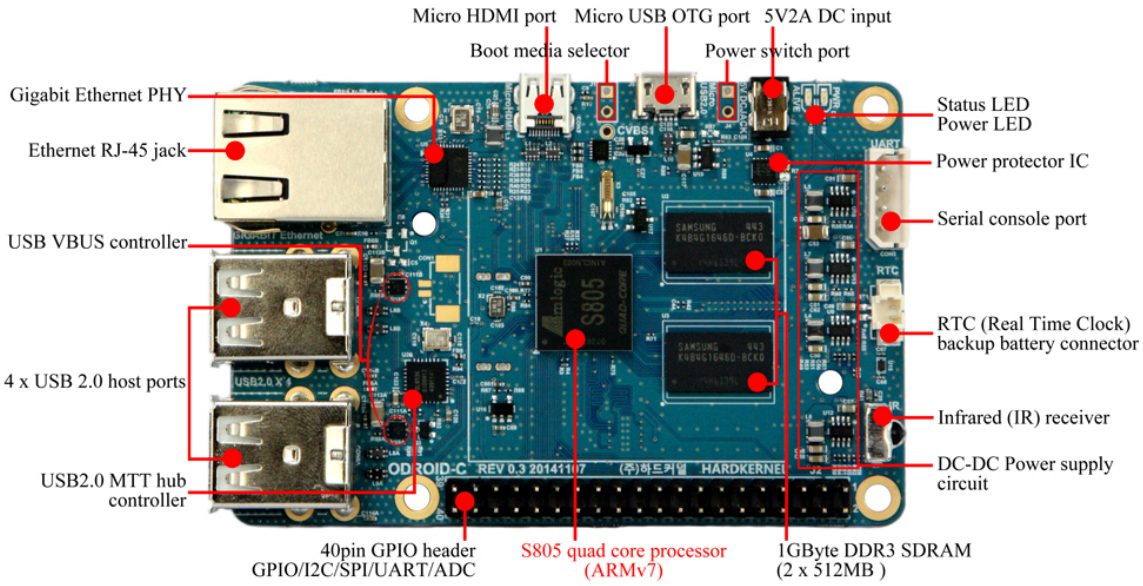

Figura 6.6: Esquema de puertos. Odroid C1

6.4.5. Procesador y memoria

Cortex-A5 a 1.5GHz. A diferencia de la Raspberry Pi, la Odroid C1 utiliza un procesador Cortex A5, que es el más pequeño y el que consume menos energía de la familia. Sin embargo, su sucesor, el Cortex A7 es aproximadamente un 20 % más potente que el A5, aun así, según diferentes análisis[3][6] hechos por la comunidad, Odroid sigue siendo más potente que Raspberry.

En cuanto a la memoria, Odroid cuenta con 1GB a 792MHz.

6.4.6. Almacenamiento

Dispone de una ranura para microSD y una para eMMC. La velocidad de lectura/escritura de las tarjetas eMMC es mucho más alta que las de las microSD, esta característica la hace perfecta para almacenar el sistema operativo, mejorando así la respuesta del mismo.

6.4.7. Periféricos y módulos

Al igual que con la Raspberry Pi, podemos utilizar todo tipo de periféricos siempre que sean compatibles con el sistema operativo que estamos utilizando.

6.4.8. Consumo y alimentación

Está alimentada por una conexión jack de 5v/2A y consume una media de 3W/4W.

6.4.9. Sistema operativo

Dispone de dos sistemas operativos oficiales.

- Android[12]
- Ubuntu[13]

Sin embargo podemos instalar otras distribuciones no oficiales linux desarrolladas por la comunidad como Fedora, Open media vault, etc.

[12] Ver enlace: `http://odroid.com/dokuwiki/doku.php?id=en:c1_release_android`
[13] Ver enlace: `http://odroid.com/dokuwiki/doku.php?id=en:c1_release_ubuntu`

7. Casos prácticos

En el siguiente capítulo vamos a llevar a cabo varios casos prácticos con los dispositivos analizados en el capítulo anterior. Nos centraremos puramente en temas de software sin entrar en hardware.

En el caso de Arduino hemos decidido no realizar ninguna prueba puesto que nos sería difícil comparar tanto su utilidad como su rendimiento frente a otros dispositivos. Sin embargo, en el siguiente link[1] dejamos a continuación unos cuantos casos donde Arduino nos sería de utilidad.

7.1. Servidor de archivos

7.1.1. Introducción

A continuación vamos a utilizar un *single-board computer* con el fin de compartir un disco duro en red para que todos los usuarios de la misma puedan acceder a su información.

Para ello utilizaremos samba, una aplicación de software libre licenciada bajo *GNU General Public License* que ofrece servicios de archivos e impresión rápidos, estables y seguros.[5]

[1] Ver enlace: `http://www.makeuseof.com/tag/10-great-arduino-projects-for-beginners/`

7.1.2. Odroid

El sistema operativo que hemos utilizado es Ubuntu 14.04.3 LTS (v1.5)[2]. Excepto cuando indiquemos lo contrario todas las órdenes se ejecutarán como usuario *root*.

Servidor:

- Primeramente conectaremos un disco duro externo mediante uno de los USB's que tiene la placa. En nuestro caso es un disco duro de 3TB.
- Ejecutaremos *fdisk -l* para identificar nuestro disco duro.

Código 7.1: Odroid. fdisk -l

```
  ~$ fdisk -l

Disk /dev/sda: 3000.6 GB, 3000592982016 bytes
255 heads, 63 sectors/track, 364801 cylinders, total 5860533168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot Start End Blocks Id System
/dev/sda1 1 4294967295 2147483647+ ee GPT
```

- Una vez hemos identificado como '/dev/sda' editamos el fichero '/etc/fstab' para que el disco duro se monte automáticamente con el sistema. Añadiremos la siguiente línea.

[2]Ver enlace: http://odroid.com/dokuwiki/doku.php?id=en:c1_ubuntu_release_note_v1.5

Código 7.2: Odroid. /etc/fstab

```
/dev/sda1 /media/storage1 ext4 defaults,auto 0 0
```

- Creamos el directorio /media/storage1 y le damos permisos al usuario *odroid* que utilizaremos más tarde.

Código 7.3: Odroid. mkdir and chown

```
~$ mkdir /media/storage1
~$ chown -R odroid.odroid /media/storage1
```

- Montamos el disco duro.

Código 7.4: Odroid. mount

```
~$ mount /dev/sda1
```

- Instalamos samba.

Código 7.5: Odroid. install samba

```
~$ apt-get -y install samba
```

- Editamos el fichero de configuración de samba /etc/samba/smb.conf[3] para añadir el directorio /media/storage1, donde previamente hemos montado nuestro disco duro /dev/sda. Añadiremos las siguientes líneas.

[3] Ver enlace: https://www.samba.org/samba/docs/man/manpages-3/smb.conf.5.html

Código 7.6: Odroid. /etc/samba/smb.conf

```
[storage1]
   comment = Storage1
   path = /media/storage1
   valid users = odroid
   writable = yes
   public = no
   guest ok = no
```

- Añadimos el usuario *odroid* del sistema como usuario de samba, escojemos una contraseña y lo habilitamos.

Código 7.7: Odroid. smbpasswd

```
~$ smbpasswd -a odroid
~$ smbpasswd -e odroid
```

- Reiniciamos samba para aplicar los cambios.

Código 7.8: Odroid. samba restart

```
~$ service smbd restart
```

Cliente:

El sistema operativo utilizado para hacer la conexión al servidor es Windows 8.1.

- Nos dirigimos a equipo y seleccionamos ”Conectar a unidad de red”

Figura 7.1: Odroid. Conectar a unidad de red

- Indicamos la direción IP del servidor y el recurso comparitdo.

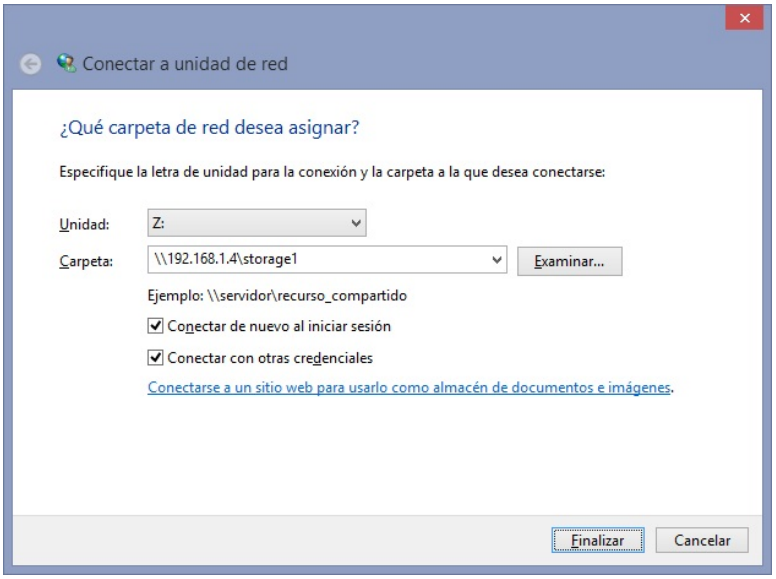

Figura 7.2: Odroid. Configuración unidad de red

- Introducimos la contraseña que hemos configurado previamente.

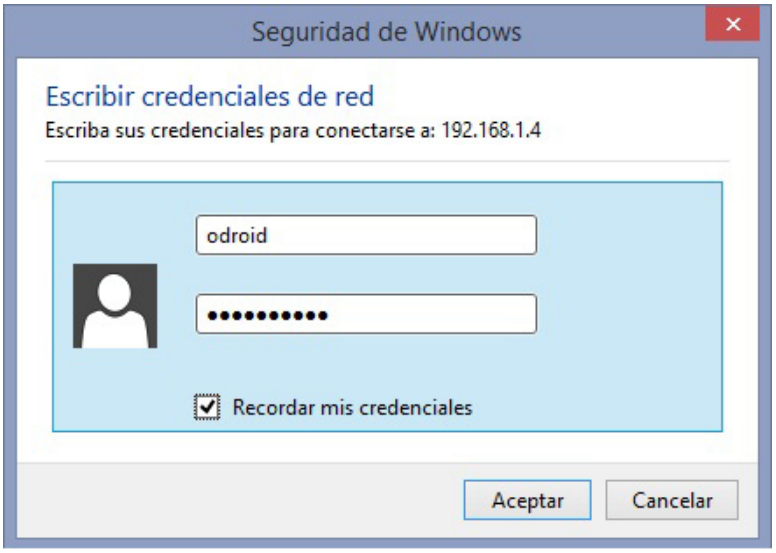

Figura 7.3: Odroid. Contraseña servidor

- Y finalmente se nos abrirá la carpeta compartida.

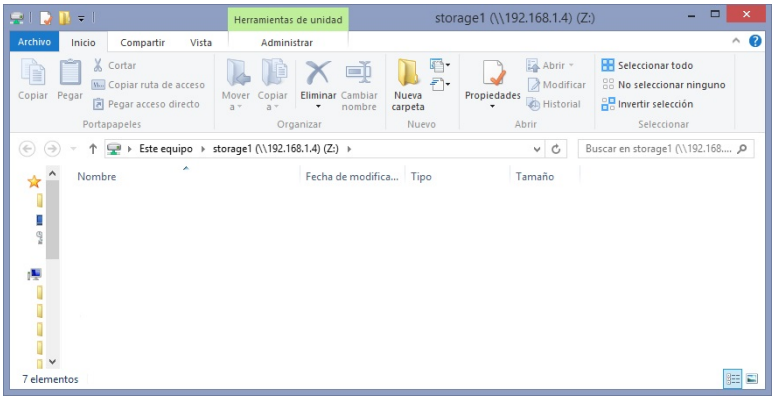

Figura 7.4: Odroid. Carpeta compartida

7.1.3. Raspberry pi

El sistema operativo que hemos utilizado es Raspbian 2015-05-05[4]. Excepto cuando indiquemos lo contrario todas las órdenes se ejecutarán como usuario *root*.

Servidor:

- Primeramente conectaremos un disco duro externo mediante uno de los USB's que tiene la placa. En nuestro caso es un disco duro de 3TB.
- Ejecutaremos *fdisk -l* para identificar nuestro disco duro.

Código 7.9: Raspberry. fdisk -l

```
  ~$ fdisk -l


Disk /dev/sda: 3000.6 GB, 3000592982016 bytes
255 heads, 63 sectors/track, 364801 cylinders, total 5860533168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

[4]Ver enlace: https://www.raspberrypi.org/downloads/raspbian/

```
Disk identifier: 0x00000000


   Device Boot Start End Blocks Id System
/dev/sda1 1 4294967295 2147483647+ ee GPT
```

- Una vez hemos identificado como '/dev/sda' editamos el fichero '/etc/fstab' para que el disco duro se monte automáticamente con el sistema. Añadiremos la siguiente línea.

Código 7.10: Raspberry. /etc/fstab

```
/dev/sda1 /media/storage2 ext4 defaults,auto 0 0
```

- Creamos el directorio /media/storage2 y le damos permisos al usuario *pi*, que vamos a utilizar más tarde.

Código 7.11: Raspberry. mkdir and chown

```
~$ mkdir /media/storage2
~$ chown -R pi.pi /media/storage2
```

- Montamos el disco duro.

Código 7.12: Raspberry. mount

```
~$ mount /dev/sda1
```

- Instalamos samba.

Código 7.13: Raspberry. install samba

```
~$ apt-get -y install samba samba-common-bin
```

- Editamos el fichero de configuración de samba /etc/samba/smb.conf[5] para añadir el directorio /media/storage1, donde previamente hemos montado nuestro disco duro /dev/sda. Añadiremos las siguientes líneas.

Código 7.14: Raspberry. /etc/samba/smb.conf

```
[storage2]
   comment = Storage2
   path = /media/storage2
   valid users = pi
   writable = yes
   public = no
   guest ok = no
```

- Añadimos el usuario *pi* del sistema como usuario de samba, escojemos una contraseña y lo habilitamos.

Código 7.15: Raspberry. smbpasswd

```
~$ smbpasswd -a pi
~$ smbpasswd -e pi
```

- Reiniciamos samba para aplicar los cambios.

Código 7.16: Raspberry. samba restart

```
~$ service samba restart
```

Cliente:

El sistema operativo utilizado para hacer la conexión al servidor es Windows 8.1.

[5]Ver enlace: https://www.samba.org/samba/docs/man/manpages-3/smb.conf.5.html

- Nos dirigimos a equipo y seleccionamos ”Conectar a unidad de red”

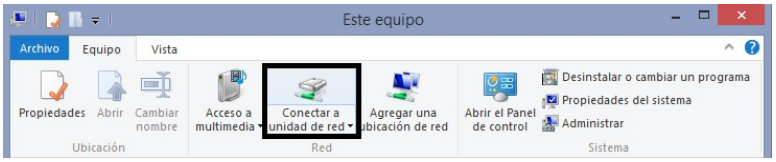

Figura 7.5: Raspberry. Conectar a unidad de red

- Indicamos la direción IP del servidor y el recurso comparitdo.

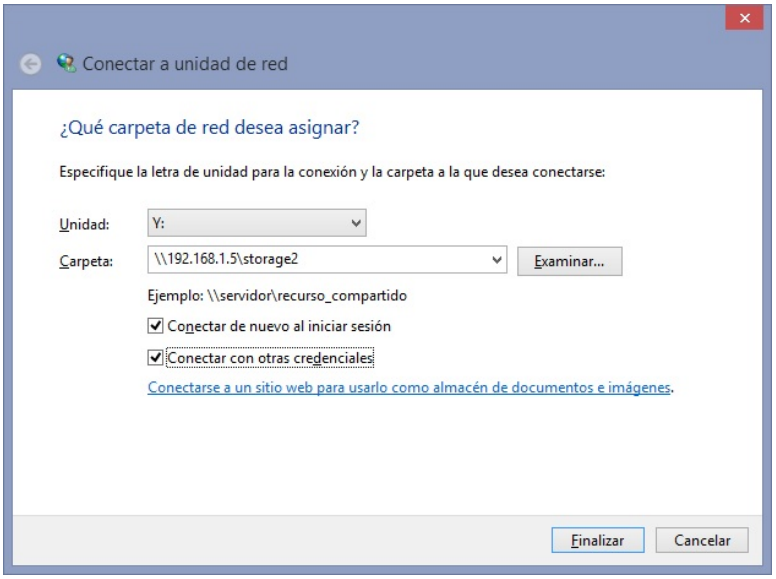

Figura 7.6: Raspberry. Configuración unidad de red

- Introducimos la contraseña que hemos configurado previamente.

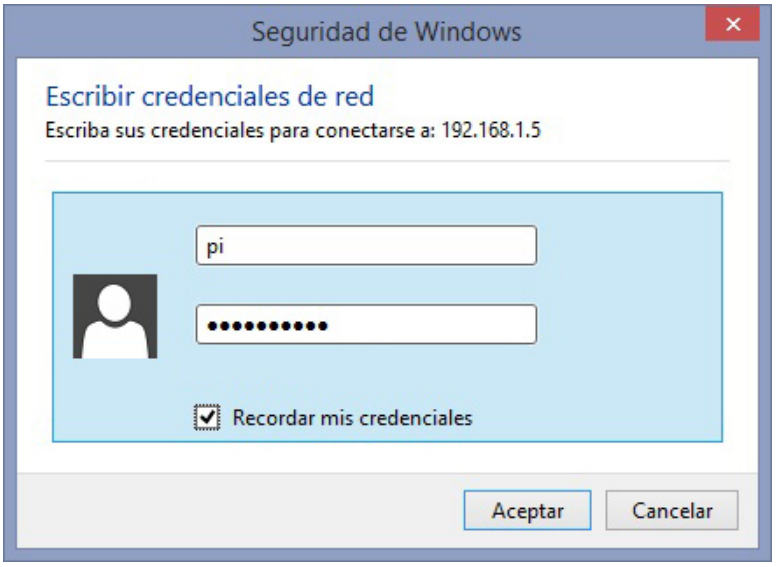

Figura 7.7: Raspberry. Contraseña servidor

- Y finalmente se nos abrirá la carpeta compartida.

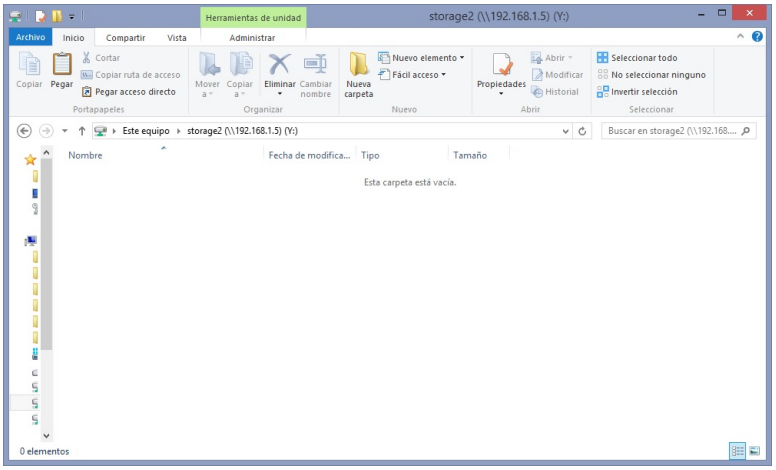

Figura 7.8: Raspberry. Carpeta compartida

7.1.4. Análisis

Con el fin de probar el rendimiento del servidor de archivos haremos la sencilla prueba de copiar un archivo y medir la velocidad de transferencia.

En el caso de Raspberry obtuvimos el siguiente resultado.

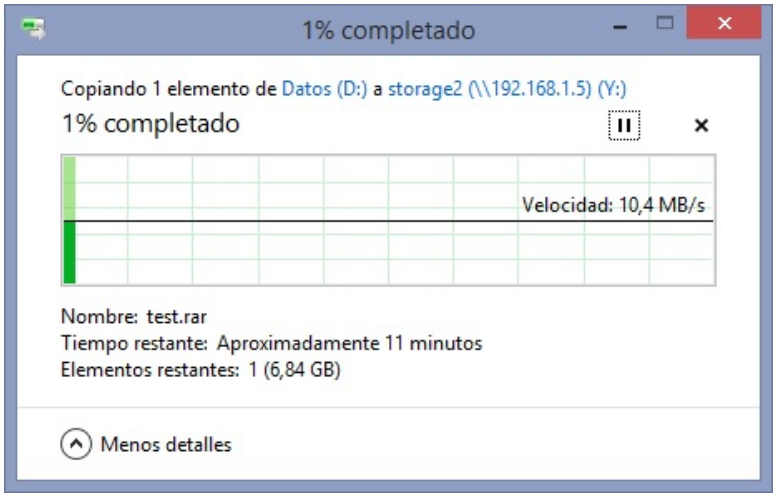

Figura 7.9: Raspberry. Prueba de velocidad

Y en cuanto a odroid obtuvimos lo siguiente.

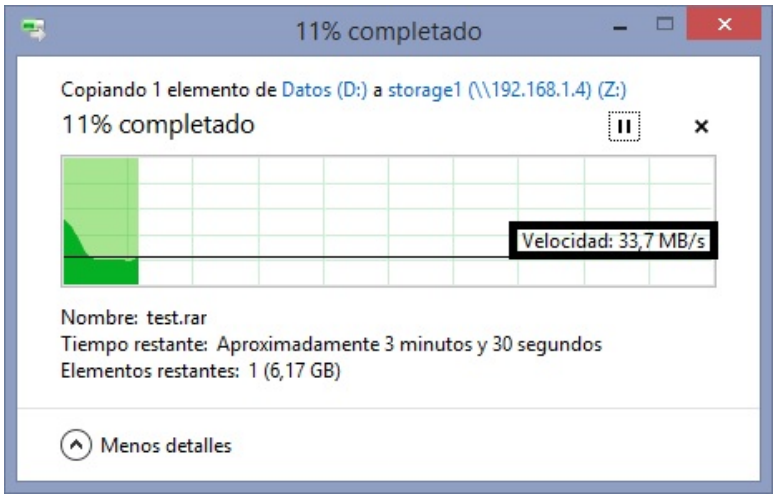

Figura 7.10: Odroid. Prueba de velocidad

El ancho de banda máximo de Odroid viene dado por el puerto USB 2.0. Es de 32mb/s aproximadamente.

En cuanto a Raspberry lo que nos limita es el puerto ethernet de 100mb/s que nos dará una velocidad máxima de 10mb/s aproximadamente.

En el caso que más de un usuario quiera acceder al mismo tiempo se dividirá la velocidad de transferencia dependendiendo de los usuarios y el ancho de banda que necesiten.

Como podemos observar, la velocidad de transferencia de Odroid es tres veces mayor que la de Raspberry. Esto es debido a que el puerto de red de 100mb/s de Raspberry hace efecto de embudo, mientras que en Odroid se puede explotar todo el ancho de banda del puerto USB 2.0 ya que la velocidad de transferencia del puerto de red es mayor.

7.1.5. Conclusión

En el caso de tener una red con pocos usuarios, o que estos no requieran un uso exhaustivo del disco compartido un *single-board computer* sería una buena solución. De lo contrario, tendríamos que irnos a dispositivos más potentes.

7.2. Servidor web

7.2.1. Introducción

En esta prueba utilizaremos el dispositivo Raspberry como servidor web y haremos alguna prueba de rendimiento con la finalidad de ver como se comporta un servidor web alojado en un *single-board computer* frente a un servidor normal y corriente.

Utilizaremos apache y apache benchmark.

Apache[6] es un servidor web open-source para sistemas operativos modernos. La meta de este proyecto es proporcionar un servidor seguro, eficiente y escalable que

[6]Ver enlace: `http://httpd.apache.org/`

cumpla con los *standards* actuales.

Apache benchmark o **ab** es una herramienta para para medir el rendimiento de un servidor apache.

7.2.2. Odroid

- Instalamos apache2 mediante el siguiente comando.

Código 7.17: Raspberry. install apache2

```
~$ apt-get install apache2
```

7.2.3. Raspberry pi

- Instalamos apache2 mediante el siguiente comando.

Código 7.18: Odroid. install apache2

```
~$ apt-get install apache2
```

7.2.4. Análisis

Para medir el rendimiento del servidor web hemos utilizado la herramienta citada en la sección 7.2.1.Introducción. El comando que hemos utilizado es el siguiente.

Código 7.19: ab command

```
~$ ab -n 1000 -c 20 <URL>
```

Este comando realiza 1000 peticiones con un máximo de 20 peticiones concurrentes a la URL indicada y proporciona unos parámetros que posteriormente hemos utilizado para construir una gráfica.

Con el fin de comparar el rendimiento de un *single-board computer* hemos realizado tres *benchmarks*, el primero con odroid, el segundo con raspberry y el tercero lo hemos realizado a un sitio web alojado en un servidor normal y corriente como es `http://www.apache.org/`.

Cabe tener en cuenta que `www.apache.org` es un sitio web online y que es muy probable que al mismo tiempo que nosotros hacíamos el *benchmark* otros usuarios estuvieran consultando y descargando ficheros. Es por esto que las pruebas de rendimiento pueden verse afectadas negativamente.

Las tres pruebas han sido realizadas a través de internet, en el caso de nuestros *single-board computers* sobre páginas muy simples alojadas en los mismos. También hemos habilitado nuestro router para que redirija las peticiones desde el exterior a través del puerto 80 al dispositivo deseado. Además, hemos utilizado un sistema de DDNS[7] con la dirección *http://jam0.asuscomm.com/* para facilitar el acceso.

[7] Ver enlace: `https://es.wikipedia.org/wiki/DNS_din%C3%A1mico`

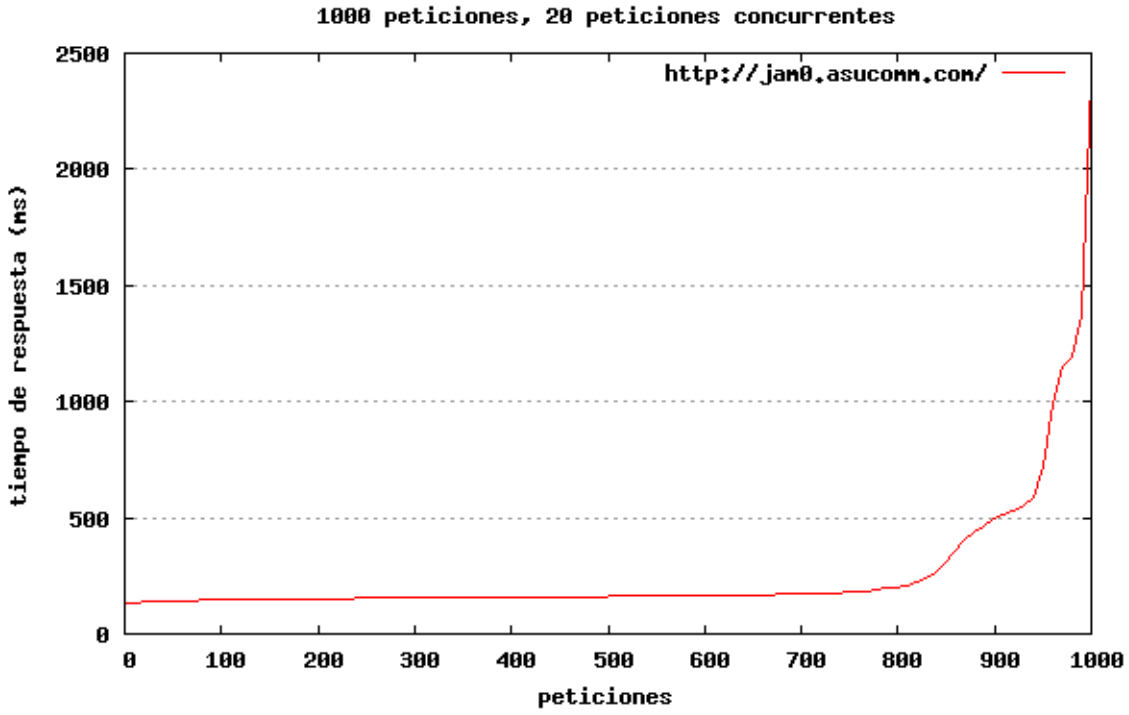

Figura 7.11: Odroid. Gráfica peticiones/ms

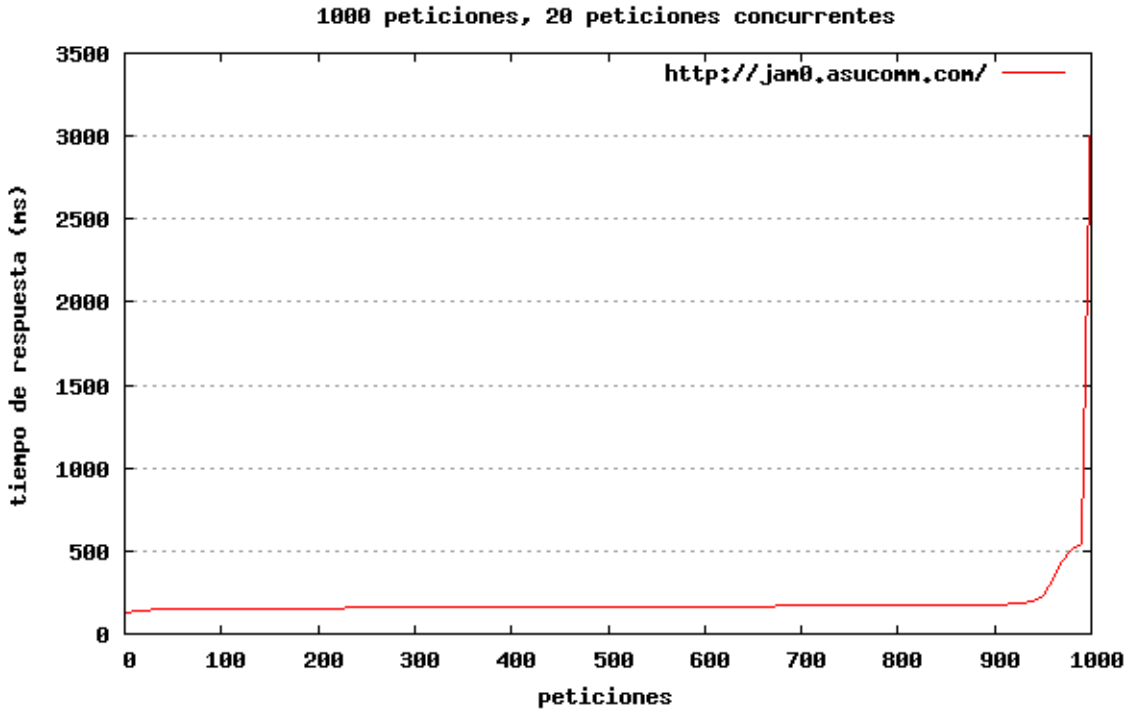

Figura 7.12: Raspberry. Gráfica peticiones/ms

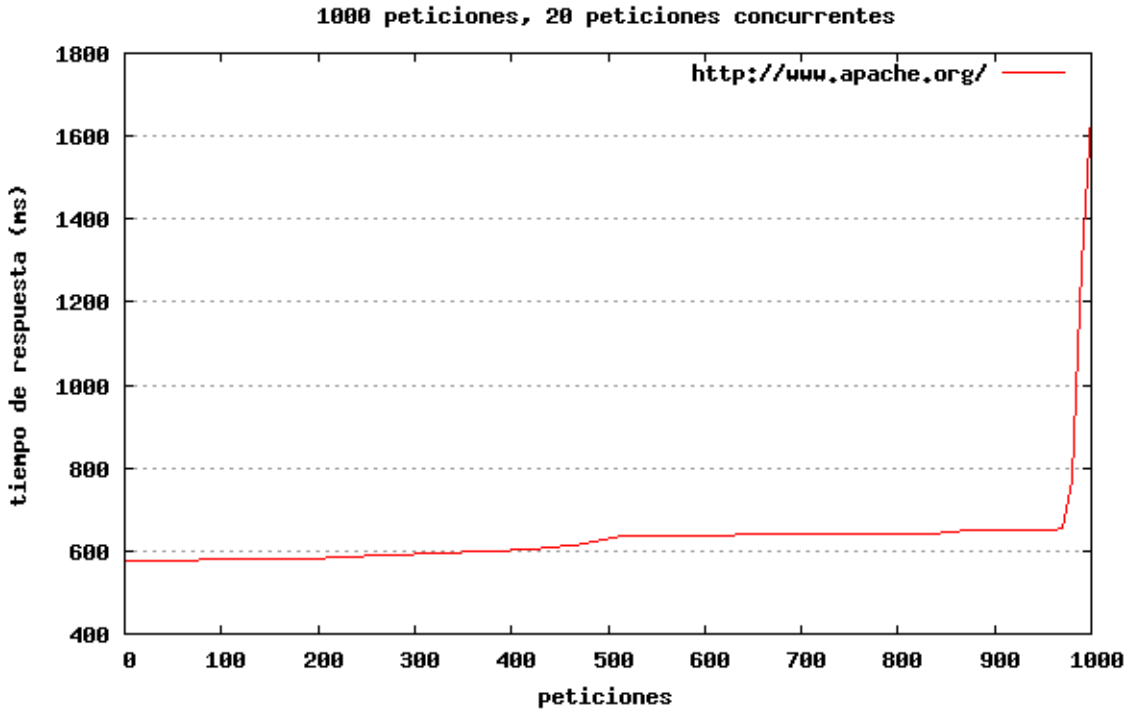

Figura 7.13: apache.org. Gráfica peticiones/ms

A partir de las gráficas sacamos las siguientes conclusiones:

Odroid

- Tarda aproximadamente 2,250 segundos en atender 1000 peticiones.
- El tiempo de respuesta aumenta exponencialmente a partir de las 825 peticiones.
- Por debajo de las 825 peticiones el tiempo promedio de respuesta es de 0.15 segundos aproximadamente.

Raspberry

- Tarda aproximadamente 3 segundos en atender 1000 peticiones.
- El tiempo de respuesta aumenta exponencialmente a partir de las 950 peticiones.
- Por debajo de las 950 peticiones el tiempo promedio de respuesta es de 0.1 segundos aproximadamente.

www.apache.org

- Tarda aproximadamente 1,6 segundos en atender 1000 peticiones.
- El tiempo de respuesta aumenta exponencialmente a partir de las 975 peticiones.
- Por debajo de las 975 peticiones el tiempo promedio de respuesta es de 0.6 segundos aproximadamente.

7.2.5. Conclusión

Concluimos que tanto Odroid como Raspberry rinden de manera muy similiar y para un sitio web simple y con pocas visitas nos funcionaría sin problemas.En caso de utilizar PHP, bases de datos, Javascripts muy pesados, etc. Un *single-board computer* no sería una buena opción.

7.3. Media center

7.3.1. Introducción

En la siguiente sección convertiremos nuestros *single-board computers* en centros multimedia que, conectados a una televisión, nos permitirán visualizar nuestro contenido multimedia de manera fácil y rápida.

El software que utilizaremos será **Kodi** en el caso de Odroid y **Openelec** en el caso de Raspberry.

Kodi[8] es un software libre y multiplataforma que permite reproducir multimedia. **Openelec**[9] es un sistema operativo destinado a ejecutar **Kodi**.

7.3.2. Odroid

Kodi viene instalado por defecto con el sistema operativo que hemos utilizado. Ubuntu 14.04.3 LTS.

7.3.3. Raspberry pi

Deberemos descargar e instalar el sistema operativo **Openelec**.

[8] Ver enlace: `http://kodi.tv/about/`
[9] Ver enlace: `http://openelec.tv/`

7.3.4. Análisis

Formatos

Kodi soporta una gran cantidad de formatos tanto de vídeo como de audio como de imágen. Entre los más populares se encuentran:

- Vídeo: avi, mkv, mpeg, mp4
- Audio: mp3
- Imagen: jpeg, bmp, gif, png

Para ver una lista completa con todos los formatos soportados consultar el siguiente enlace[10].

Conectividad

En cuanto a conectividad, alguna de las características básicas del sistema son las siguientes:

- Posibilidad de acceder a contenido en internet. Youtube[11], TwitchTV[12]...
- Posibilidad de conectar periféricos mediante los USB's. Teclado, ratón, adaptador de red inalámbrico...
- Posibilidad de conectar dispositivos de almacenamiento externos y reproducir multimedia de los mismos.
- Posibilidad de controlar Kodi a través de la app *Oficial Kodi Remote*

[10]Ver enlace: `http://kodi.wiki/view/Features_and_supported_formats`
[11]Ver enlace: `https://www.youtube.com/`
[12]Ver enlace: `http://www.twitch.tv/`

Rendimiento

Para medir el rendimiento de nuestro *single-board computer* hemos habilitado la visualización de recursos mientras reproduciamos un archivo de vídeo en formato *1080p*[13].

De todas las tareas que podemos realizar en Kodi, la más pesada o costosa en cuanto a recursos es la de reproducir vídeo, tanto de internet como de un medio local. Esto es debido a que cuando reproducimos vídeo, el dispositivo tiene que procesar cada uno de los fotogramas. Además también tiene que sincronizar el audio. Es por eso que el reproducir un vídeo de alta calidad consumirá más recursos que reproducir uno de baja calidad, ya que tiene que procesar o bien más fotogramas, o bien fotogramas de mayor calidad.

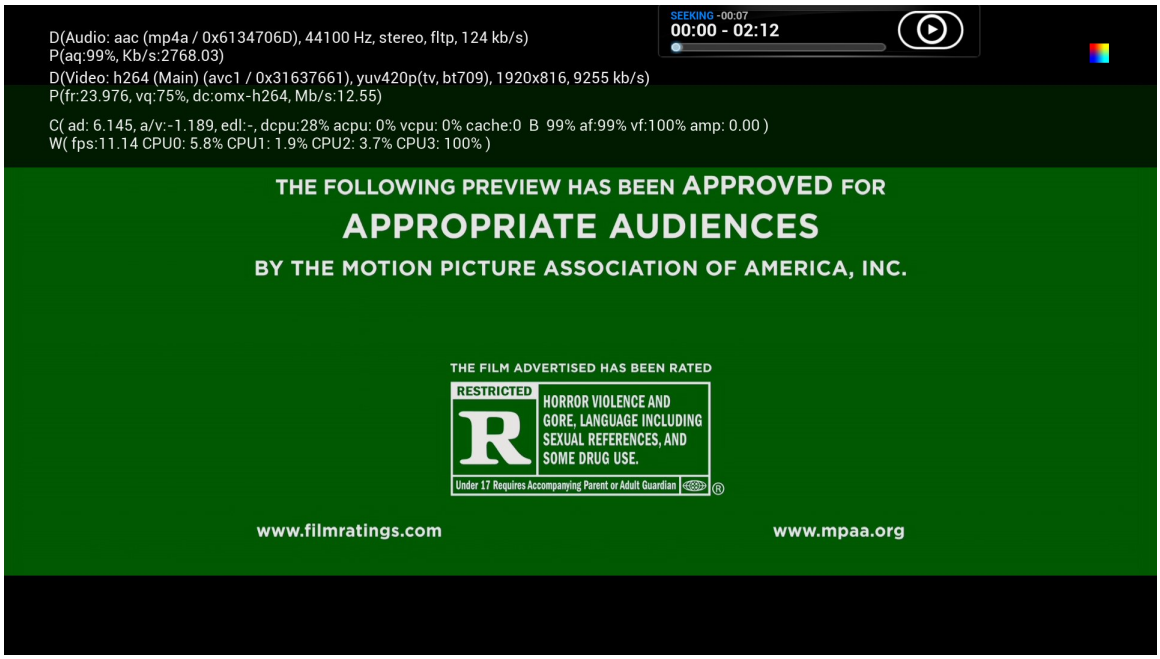

Figura 7.14: Kodi. Rendimiento reproductor

Como vemos en la figura 7.14.Kodi. Rendimiento reproductor, la CPU3 se muestra a un 100 % de carga, mientras que las otras tres CPU's (CPU0, CPU1 y CPU2)

[13] Ver enlace: `https://es.wikipedia.org/wiki/1080p`

están por debajo del 6 %. Esto significa que estamos utilizando aproximadamente un 30 % del procesador, por tanto asumimos que el hardware está más que preparado para desarrollar este tipo de acciones sin problema.

Interfaz

Al arrancar Kodi por primera vez encontraremos el siguiente menú. Donde entre otras opciones encontraremos pictures, videos y music.

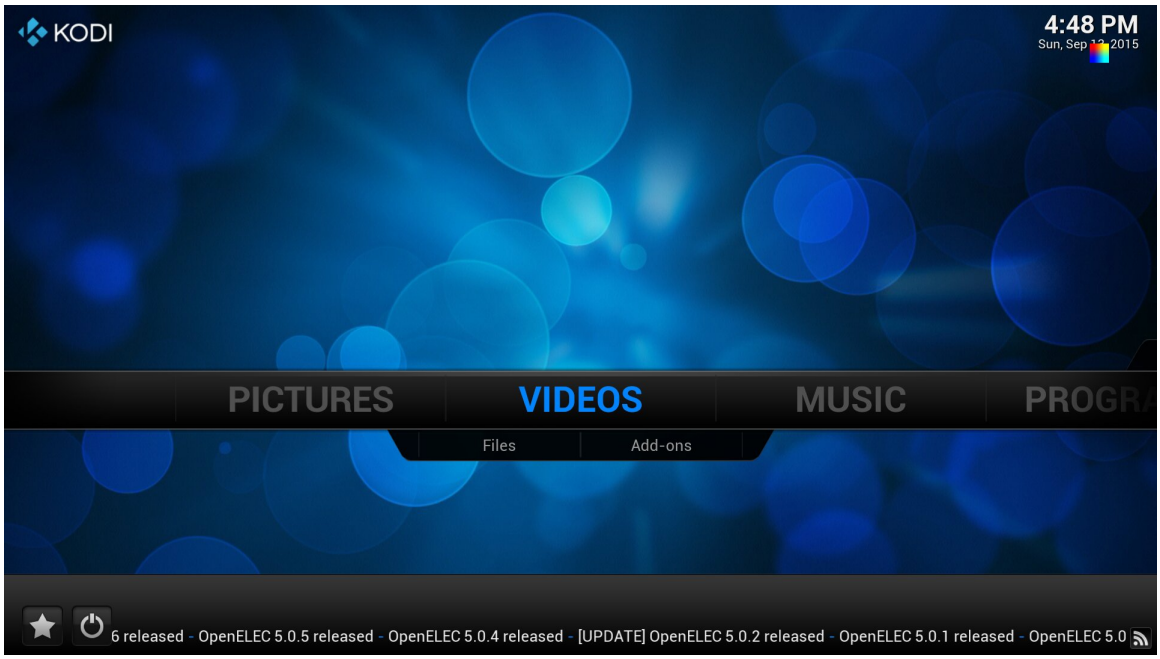

Figura 7.15: Kodi. Menú

Visualizar imágenes

Para visualizar nuestras imágenes accederemos, mediante el menú principal (7.15.Kodi. Menú), al apartado pictures.

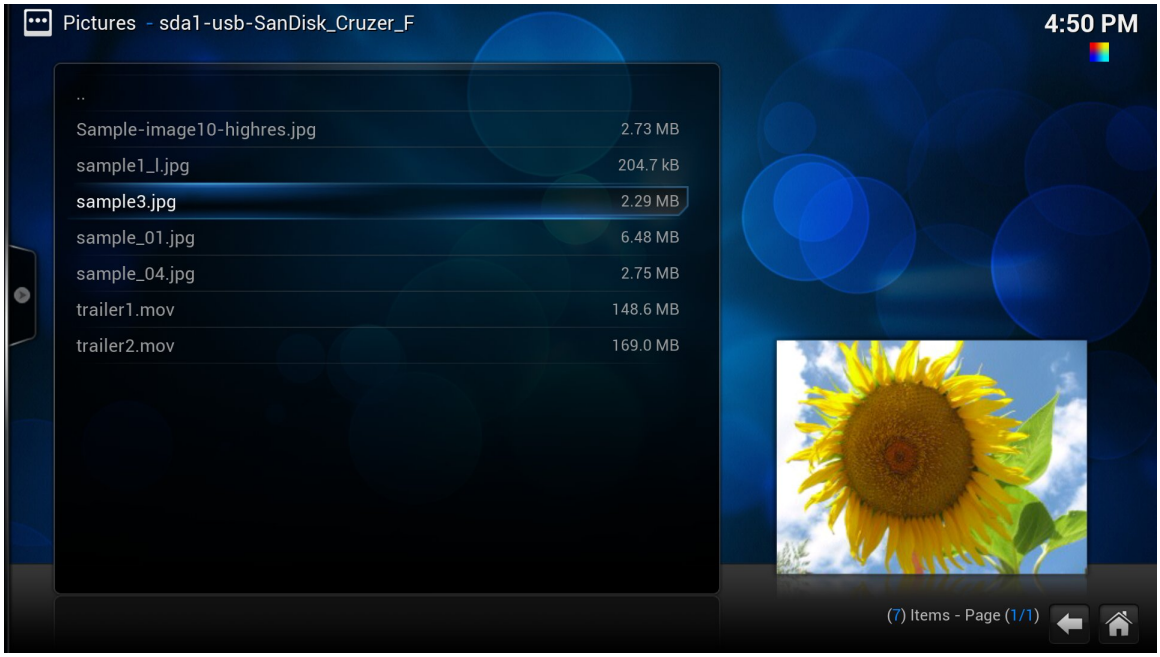

Figura 7.16: Kodi. Pictures

Visualizar vídeos

Para visualizar nuestros vídeos accederemos, mediante el menú principal (7.15.Kodi. Menú), al apartado vídeos.

Figura 7.17: Kodi. Videos

Y, una vez en la lista de ficheros a reproducir seleccionaremos cualquiera de ellos para reproducirlos.

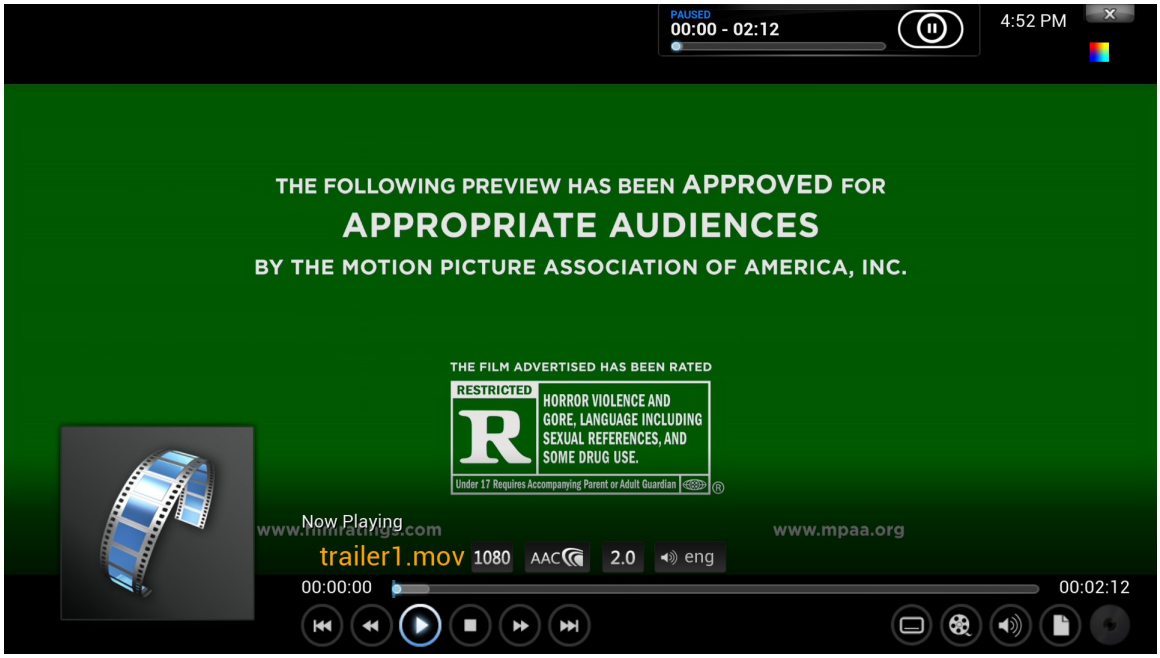

Figura 7.18: Kodi. Reproductor

Reproducir contenido de internet

La reproducción de contenido de internet de Kodi funciona mediante un sistema de *add-ons* que pueden ser descargados de los repositorios que vienen por defecto con el sistema o de internet. Una vez descargados e instalados nos ofrecen acceso a plataformas multimedia como YouTube, TwitchTV, etc.

Por defecto, Kodi, no incorpora ningún *add-on*.

En la siguientes figuras se muestran el *add-on* de YouTube y la reproducción de un vídeo.

Figura 7.19: Kodi. Video add-ons

Figura 7.20: Kodi. Youtube Video

De igual manera podemos descargar e instalar *add-ons* de música.

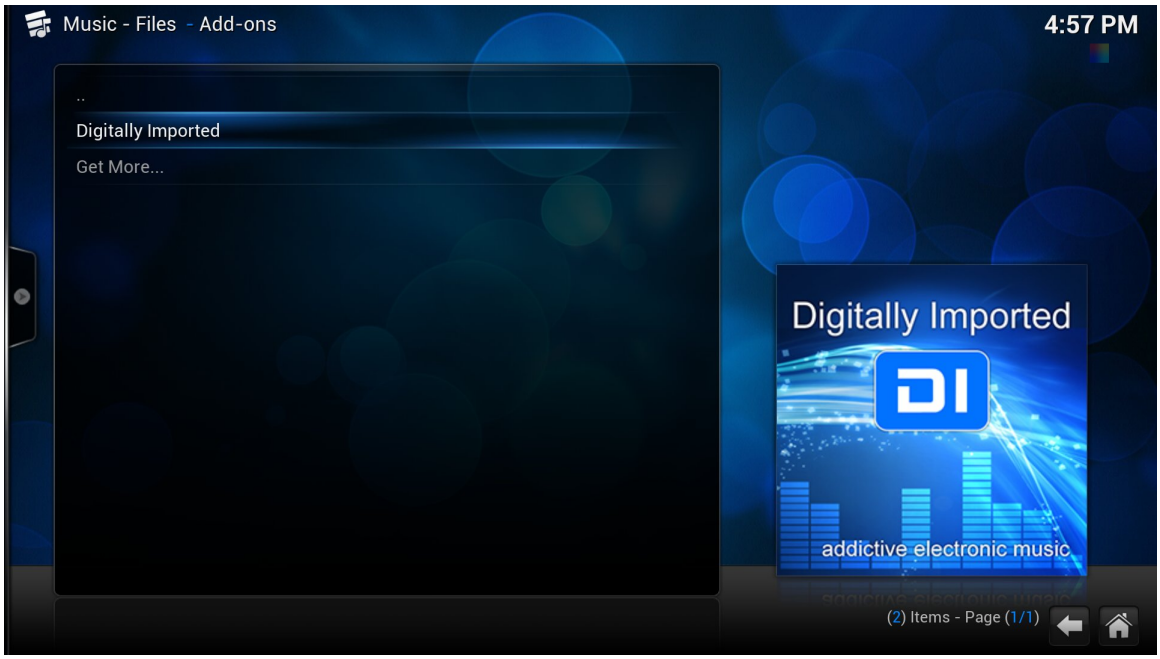

Figura 7.21: Kodi. Music add-ons

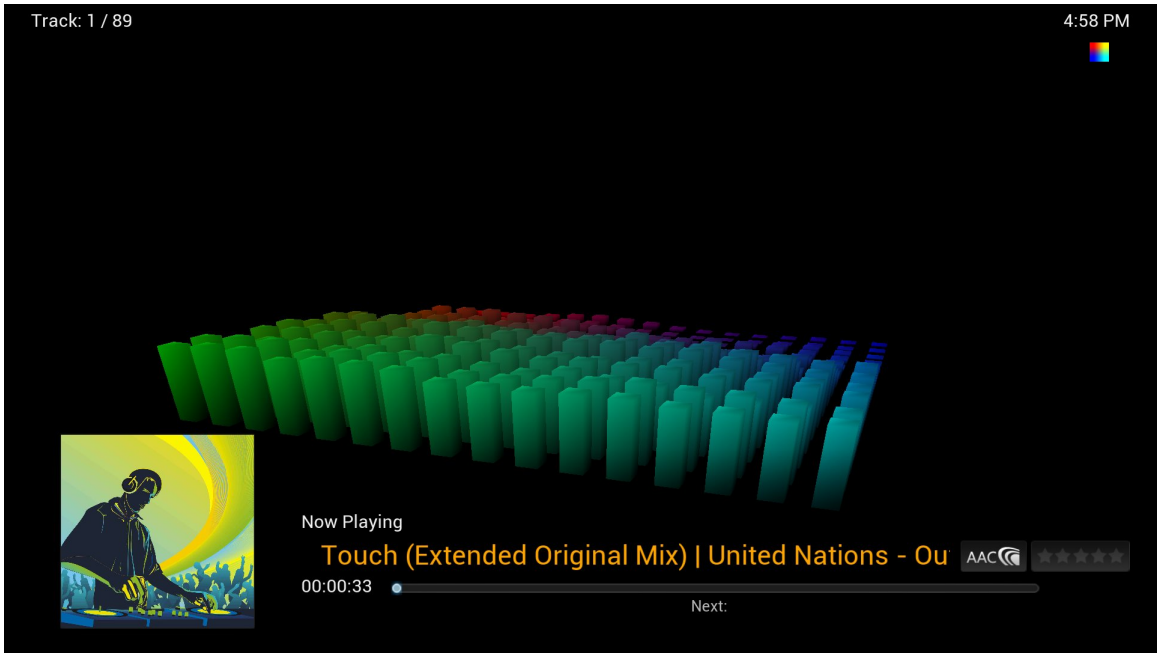

Figura 7.22: Kodi. Music streaming

7.3.5. Conclusión

Tanto Odroid como Raspberry funcionan a la perfección reproduciendo imágenes, vídeo y música. No hemos experimentado ningún corte y los FPS's[14] han sido constantes en todo momento.

De todas los casos prácticos que hemos realizado este es el que nos ha parecido de más utilidad para un *single-board computer* ya que mediante un dispositivo pequeño, que cabe en la palma de la mano, y que puede ser alimentado desde un puerto USB que, a día de hoy todas las televisiones modernas incorporan, podemos tener un centro multimedia que para nuestra opinión está a años luz del sistema de las conocidas SmartTV[15].

[14] FPS: Fotogramas por segundo

[15] Ver enlace: http://www.which.co.uk/reviews/televisions/article/what-is-smart-tv

8. Conclusiones

Para acabar podemos decir que el resultado del proyecto ha sido el esperado, no obstante, de haber tenido más tiempo podríamos haber hecho un análisis más exhaustivo de los dispositivos además de haber incluido más.

Como resultado del proyecto hemos visto las diferentes arquitecturas de computador que se emplean hoy en día. También hemos conocido los microcontroladores y *single-board computers* así como algunas de sus aplicaciones y limitaciones.

A medida que avanza la tecnología encontraremos *single-board computers* cada vez más pequeños y potentes que permitiran realizar ciertas tareas igual o mejor que ordenadores de sobremesa o portátiles tal y como los conocemos hoy en día. Todo esto en una placa de tamaño muy reducido, que apenas se calienta, que no hace ruido y que consume una ínfima parte de lo que consume un ordenador normal y corriente.

Por tanto, concluimos que en un futuro los *single-board computers* jugarán un papel importante en el ámbito de la tecnología.

Bibliografía

[1] digital.ni.com. What is a pulse width modulation? http://digital.ni.com/public.nsf/allkb/294E676237526566862 56DB800508989, consultado en 2015.

[2] egr.msu.edu. *General purpose Input/Output (GPIO)*. http://www.egr.msu.edu/classes/ece480/capstone/fall09/group03/AN_balachandran.pdf, consultado en 2015.

[3] HardKernel. Análisis hechos por hardkernel. http://www.hardkernel.com/main/products/prdt_info.php?g_code=G141578608433&tab_idx=2, consultado en 2015.

[4] importancia.org. Arquitectura arm. http://www.importancia.org/arquitectura-arm.php, consultado en 2015.

[5] samba.org. Samba. https://www.samba.org/, consultado en 2015.

[6] silabs.com. *Which ARM Cortex core is right for your application*. http://www.silabs.com/Support%20Documents/TechnicalDocs/Which-ARM-Cortex-Core-Is-Right-for-Your-Application.pdf, consultado en 2015.

[7] Wikipedia. Arquitectura arm. http://es.wikipedia.org/wiki/Unidad_central_de_procesamiento, consultado en 2015.

[8] Wikipedia. Dispositivos de entrada/salida. http://es.wikipedia.org/wiki/Perif%C3%A9rico_de_Entrada/Salida, consultado en 2015.

[9] Wikipedia. Memoria de acceso aleatorio. http://es.wikipedia.org/wiki/Memoria_de_acceso_aleatorio, consultado en 2015.

[10] Wikipedia. Unidad de procesamiento central. http://es.wikipedia.org/wiki/Unidad_central_de_procesamiento, consultado en 2015.