

Diseño y despliegue de un clúster de bajo presupuesto para el desarrollo de las prácticas de PSD

Daniel Quiñones Sánchez
Miguel Romero Martínez

Grado en Ingeniería de Computadores
Facultad de Informática



Universidad Complutense de Madrid

Curso Académico 2017/2018

Directores:

Alberto Núñez Covarrubias
Luis Llana Díaz

Agradecimientos



A mi compañero y amigo Rome, por aguantarme en todo momento y realizar la mayor parte del trabajo.

Daniel



A mi compañero y amigo Daniel, por aguantarme en todo momento y realizar la mayor parte del trabajo.

Miguel



Daniel && Miguel

Índice general

Índice de figuras	v
Abstract	vii
1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
2. Componentes	3
2.1. Crowfounding	3
2.2. Raspberry Pi Modelo B	4
2.2.1. Especificaciones	4
2.2.2. Límites térmicos	5
2.2.3. Disipadores de calor	6
2.3. Switch D-LINK DGS-1008D	6
2.3.1. Especificaciones	7
2.4. Tarjeta microSD SanDisk	7
2.5. Ventiladores	8
2.6. Conversor USB 3.0 a Ethernet	9
2.6.1. Especificaciones	9
2.7. Cargador USB Vs hub USB	10
3. Configuración del clúster	11
3.1. Virtualización del sistema	11

3.2.	Sistema centralizado	12
3.3.	Arquitectura del Clúster	13
3.4.	Configuración de la red	14
3.4.1.	Configuración de DHCP	15
3.4.2.	Hostname	17
3.4.3.	Network File System (NFS)	18
3.4.4.	Instalación de NFS	18
3.4.5.	Creación y ejecución del servidor NFS como un <i>daemon</i> del sistema .	19
3.5.	Instalación de OMNeT++, INET y SIMCAN	20
3.5.1.	Instalación de OMNeT++	21
3.5.2.	Instalación de INET	21
3.5.3.	Instalación de SIMCAN	22
3.5.4.	Problemas	22
3.6.	Configuraciones derivadas de la arquitectura	22
3.6.1.	Modificación del GRUB	23
3.6.2.	Habilitar arranque automático de un usuario	23
3.6.3.	Forzar el arranque sin HDMI	24
3.7.	Seguridad	24
3.8.	Eliminar usuarios y permisos	24
4.	Diseño de Prototipos	25
4.1.	Características del modelado	25
4.2.	Modelo 1	26
4.2.1.	Resultado	27
4.3.	Modelo 2	28
4.3.1.	Resultado	29

4.4. Modelo 2b	30
4.5. Modelo 3	31
4.5.1. Modelado 3D	31
4.5.2. Resultado	32
5. Tests de Diseño	33
5.1. Experimental Settings	33
5.2. Prueba 1	34
5.2.1. Escenario	34
5.2.2. Resultados	34
5.2.3. Conclusiones	35
5.3. Prueba 2	36
5.3.1. Escenario	36
5.3.2. Resultados	36
5.3.3. Conclusiones	37
5.4. Prueba 3	38
5.4.1. Escenario	38
5.4.2. Resultados	38
5.4.3. Conclusiones	39
5.5. Prueba 4	40
5.5.1. Escenario	40
5.5.2. Resultados	40
5.5.3. Conclusiones	40
5.6. Conclusiones generales	41
6. Cliente Servidor en JAVA	43
6.1. Creación de un .jar	43

7. Modelo Final	45
7.1. Diseño Modelo	46
7.1.1. Diseño 3D	46
7.1.2. Resultado Modelo	47
7.2. Pruebas Modelo	47
7.2.1. Resultados Pruebas	47

Índice de figuras

2.1.	Raspberry Pi Modelo 3 b	4
2.2.	Captura Térmica	5
2.3.	Disipadores	6
2.4.	Switch D-LINK DGS-1008D	7
2.5.	microSD SanDisk	7
2.6.	Ventiladores USB	8
2.7.	Conversor USB 3.0 a Ethernet	9
2.8.	Cargador USB	10
3.1.	Sistema de archivos	12
3.2.	Sistema centralizado	13
3.3.	Esquema	14
3.4.	Servidor DHCP	15
4.1.	Modelo 1 3D	26
4.2.	Resultado Modelo 1	27
4.3.	Modelo 2 3D	28
4.4.	Resultado Modelo 2	29
4.5.	Modelo 2b 3D	30
4.6.	Modelo Modelo 3 3D	31
4.7.	Resultado Modelo 3	32
5.1.	Prueba 1, Modelo 1	34
5.2.	Prueba 1, Modelo 2	35

5.3. Prueba 1, Modelo 2b	35
5.4. Prueba 1, Modelo 3	35
5.5. Prueba 2, Modelo 1	36
5.6. Prueba 2, Modelo 2	36
5.7. Prueba 2, Modelo 2b	37
5.8. Prueba 2, Modelo 3	37
5.9. Prueba 3, Modelo 1	38
5.10. Prueba 3, Modelo 2	38
5.11. Prueba 3, Modelo 2b	39
5.12. Prueba 3, Modelo 3	39
5.13. Prueba 4, Prueba de estrés	40
7.1. Modelo Final 3D	46
7.2. Resultado Modelo Final	47
7.3. Resultados Prueba 1	47
7.4. Resultados Prueba 2	48
7.5. Resultados Prueba 3	48

Abstract

*;Tranquilos, es sólo un nombre! Como la Zona
de la Muerte o la Zona sin Retorno. Esos
nombres son normales en la Galaxia del Terror.*

Profesor Hubert Farnsworth

The main objective of this work is the development of a low budget computing cluster. For this we have made use of the so-called Raspberry Pi reduced-board computers, as processing nodes on a Debian Jessie Linux distribution.

We set out the following specific objectives: design of the container, distribution of each of the elements within it, study of the temperatures and behavior of the hardware under stress situations, deployment and development of the system software, performance study of the same, generation of installation guides and specific aspects of system configuration, development of a task planner for the distribution of work between the nodes and development of software to send and receive jobs to the cluster.

Throughout the document, each of the previous points will be developed. This work is intended to complement the laboratory practices that are carried out in the subject of Distributed Systems Programming (PSD) in the Computing Faculty of the Complutense University of Madrid.

At the end of the project, the cluster will allow to students the possibility of performing some of the computation tasks on it, will also serve as an example of a real distributed system, with which to strengthen the knowledge of the subject.

Capítulo 1

Introducción

However difficult life may seem, there is always something you can do, and succeed at. It matters that you don't just give up.

Stephen William Hawking

Este trabajo consiste en el desarrollo de un clúster de cómputo de bajo presupuesto destinado a complementar las prácticas de laboratorio que se realizan en la asignatura de *Programación de Sistemas Distribuidos* (PSD). Permitirá a los alumnos la posibilidad de realizar algunas de las tareas de cómputo sobre él. Servirá además como ejemplo de un sistema distribuido real, con el que poder afianzar los conocimientos de la asignatura.

1.1. Motivación

En los últimos años se ha incrementado los proyectos enfocados a construir sistemas distribuidos utilizando hardware de bajo coste. Generalmente, estos son llevados a cabo por universidades e iniciativas de particulares. Principalmente estos se han centrado en aumentar el número de dispositivos en cada proyecto, pero no existe un estudio de comportamiento real de uno de ellos, así como una guía que establezca los pasos a seguir, nuestro propósito ha sido el de reflejar todos los aspectos del desarrollo de uno de estos clusteres, centrándonos particularmente en la correcta disposición y distribución de los componentes para

mejorar aspectos como la accesibilidad, la correcta refrigeración de los nodos y la mejora del rendimiento en ellos.

Existen multitud de proyectos de este tipo, destacaremos los más interesantes:

- [VMW Research Group Raspberry Pi Cluster](#), dispone de un clúster de 24 nodos realizado con raspberry Pi 2 con una interfaz de pantalla táctil y dos adaptadores de Ethernet que controlan la fuente de alimentación , sirve DHCP, NFS.
- [Universidad de Southampton](#), investigadores de esta universidad han construido una supercomputadora de Raspberry Pi unidas con Lego. El profesor Simon Cox y su equipo construyeron la supercomputadora de 64 procesadores y 1 TB de memoria. Tiene un coste aproximado de 3100 euros.
- [David Guill](#), en su web Like Magic Appears ofrece una guía de construcción de un clúster de 40 nodos y dispone de material audiovisual como guía.

1.2. Objetivos

Nos planteamos los siguientes objetivos específicos: diseño del contenedor, distribución de cada uno de los elementos dentro de éste, estudio de las temperaturas y comportamiento del Hardware bajo situaciones de estrés, despliegue y desarrollo del software del sistema, estudio de rendimiento del mismo, generación de guías de instalación y aspectos específicos de configuración del sistema, desarrollo de un planificador de tareas para la distribución de trabajos entre los nodos y desarrollo de un software para realizar el envío y recepción de trabajos al clúster.

Capítulo 2

Componentes

*Bueno, pero aparte del alcantarillado, la sanidad,
la enseñanza, el vino, el orden público, la
irrigación, las carreteras y los baños públicos,
¿qué han hecho los romanos por nosotros?*

The Life Of Brian

A lo largo de este capítulo hablaremos de cada uno de los elementos que componen el cluster, detallando alguna de sus características mas importantes y evaluando los problemas derivados de estas.

2.1. Crowfounding

Cada uno de los componentes que se describen a continuación fueron obtenidos a través del Proyecto de Innovación Docente de la Universidad Complutense. Una vez finalizado el proyecto estos formarán parte del material accesible a los alumnos para futuros trabajos. Con el presupuesto obtenido para este proyecto, aproximadamente setecientos cincuenta euros, hemos obtenido material suficiente para el montaje final.

2.2. Raspberry Pi Modelo B

Hemos elegido Raspberry Pi Modelo B como nodo de cómputo debido a su limitado consumo, tamaño reducido y bajo coste. Las características de ésta en cuanto a número de procesadores, velocidad de éstos, así como, tamaño de memoria *RAM*, hace que sea idónea para el diseño de clusteres de cómputo.



Figura 2.1: Raspberry Pi Modelo 3 b

2.2.1. Especificaciones

- Chipset Broadcom BCM2387
- 1,2 GHz de cuatro núcleos ARM Cortex-A53
- 1 GB LPDDR2
- Ethernet socket Ethernet 10/100 BaseT
- 802.11 LAN inalámbrica
- HDMI rev 1.3 y 1.4
- USB 4x Conector USB 2.0
- Conector GPIO

2.2.2. Límites térmicos

Uno de los mayores problemas de utilizar Raspberry Pi para el proyecto es que esta carece de elementos de disipación de calor integrados, lo que hace que sea especialmente sensible a las altas temperaturas. Bajo una situación de poco estrés mantiene unos rangos de temperatura estables, sobre los 40°C, sin embargo, en condiciones de carga de trabajo, el aumento de la temperatura es elevado, llegando a su punto crítico sobre temperaturas cercanas a los 80°C, condiciones en las cuales, por seguridad, se produce un apagado súbito del sistema. En la figura 2.3 se puede apreciar un procesador con sobre la temperatura crítica de apagado.

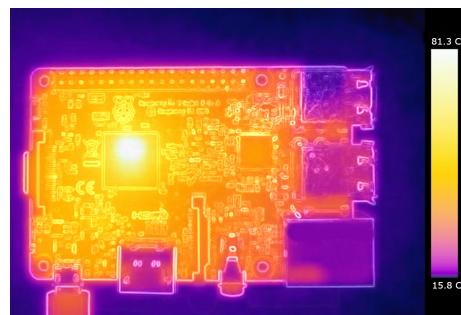


Figura 2.2: Límite térmico de la Raspberry

Los elementos que se ven especialmente afectados son memoria, controlador de Ethernet y el procesador.

Debido a la disposición de cada Raspberry dentro del clúster, en forma de columna, existe una gran proximidad entre ellas. Aquellas que se encuentran en la parte central de la estructura estén expuestas a un efecto de *tubo de calor* proveniente de los nodos próximos a ellas. Por esto es necesario disponer de un buen método de disipación de calor, a fin de evitar el sobrecalentamiento del sistema. Tanto la disposición, como otras soluciones aportadas a este problema se describen mas ampliamente en el capítulo 4, sin embargo, a fin de lidiar con este problema incluimos algunos disipadores de calor disponibles para este modelo de Raspberry.

2.2.3. Disipadores de calor

Este componente contiene tres disipadores diferentes, todos ellos de aluminio, con medidas específicas para el procesador, memoria y controlador Ethernet de una RaspberryPi.



Figura 2.3: Disipadores

Su un precio muy reducido y, como veremos más adelante, aunque por si solos no suponen una gran mejora, aproximadamente unos 2°C o 3°C menos que una Raspberry que no disponga de ellos. Con una buena ventilación ofrecen una mejora realmente notoria.

2.3. Switch D-LINK DGS-1008D

La elección del switch es importante, ya que, limita el número de nodos que puede haber dentro del clúster. Esto, junto con los cables de alimentación, tiene una influencia directa sobre el tamaño del contenedor, así como en la escalabilidad. Por otro lado, el presupuesto disponible, no nos ha permitido disponer de más nodos de cómputo. Debido a todos estos parámetros decidimos trabajar con un switch de ocho puertos, lo que permite disponer de hasta siete nodos de cómputo, ya que una de las entradas del mismo sirve como puente entre módulos, permitiendo, en caso de que se requiera, aumentar el número de nodos aumentando el número de contenedores.



Figura 2.4: Switch D-LINK DGS-1008D

2.3.1. Especificaciones

- 8 Puertos Ethernet 1000/100/10 Mbps
- Velocidad de transferencia: 2000 Mbps full duplex

Este modelo permite alcanzar una velocidad de hasta 1000 Mbps, sin embargo el modelo de Raspberry Pi utilizado está limitado a 100 Mbps. En la versión mas reciente de Raspberry (Modelo B+), este límite sigue vigente, sin embargo se espera que posteriores modelos alcancen esta velocidad, lo que mejoraría las comunicaciones dentro del equipo.

2.4. Tarjeta microSD SanDisk

Raspberry Pi no lleva incorporado almacenamiento en disco, en vez de eso, dispone de una ranura microSD en la cual se instala el sistema operativo.



Figura 2.5: microSD SanDisk

La capacidad de las tarjetas es un factor a tener en cuenta, ya que, la instalación del sistema operativo en tarjetas superiores a los 32 Gigabytes requiere de un software específico. Como hemos podido comprobar, existen incompatibilidades entre las versiones del sistema operativo y el software de instalación para algunas versiones de Raspbian Jessie.

Debido a que el sistema al completo no supera los *11 Gigabytes* tanto en los nodos esclavo como en el maestro decidimos usar tarjetas de *16 Gigabytes* para todos los nodos de la red, de esta forma reducimos el coste total además de evitar problemas de compatibilidad con algunas versiones de Raspbian.

2.5. Ventiladores

Además de los disipadores de calor disponemos de ventiladores. Disponen de una fuente de alimentación *USB* y tienen un tamaño de *120mm* el cuál se adapta perfectamente al número de Raspberrys que componen cada torre.

Como desarrollaremos más adelante en el capítulo miau, el uso de ventiladores, y principalmente la disposición de éstos es esencial para la refrigeración del recipiente.



Figura 2.6: Ventiladores USB

2.6. Conversor USB 3.0 a Ethernet

Raspberry 3 modelo B posee un único puerto Ethernet, el nodo maestro necesita un puerto extra, dedicado al front-end y el back-end. Éste dispositivo permite ampliar el número de puertos Ethernet.

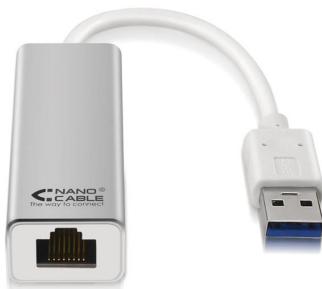


Figura 2.7: Conversor USB 3.0 a Ethernet

2.6.1. Especificaciones

- Conexión USB 3.0
- Compatible con IEEE 802.3, 802.3u y 802.3ab
- Chipset RTL8153
- Velocidad de transferencia de datos: 10, 100, 1000 Mbps
- Detección de crossover y corrección automática

Debido a sus características puede alcanzar una tasa de transferencia de *1000 Mbps*, a diferencia del puerto Ethernet instalado en la Raspberry Pi, esto permite disponer de una mayor velocidad de cara al front-end, permitiendo unas comunicaciones mas fluidas con el exterior del clúster.

2.7. Cargador USB Vs hub USB

La entrada principal de energía de Ramberry Pi es una entrada *micro USB*, aunque a través de su puerto *GPIO* puede conectarse a través de una entrada de $5V^1$. Sin embargo, esta tensión es insuficiente para el propósito del proyecto, por lo que es necesario mantener una alimentación de $3A^2$ para realizar correctamente las tareas de computo.



Figura 2.8: Cargador USB

Inicialmente dispusimos de un *hub USB* conectado a red, con ocho puertos en los cuales conectábamos tanto las Raspberries como los ventiladores. El reparto de energía de este dispositivo resultó insuficiente y optamos por dos cargadores de cuatro entradas, que repartían de una manera mas efectiva su conexión a la red eléctrica.

¹Voltios

²Amperios

Capítulo 3

Configuración del clúster

Tenemos que fabricar máquinas que nos permitan seguir fabricando máquinas, porque lo que no va a hacer nunca la máquina es fabricar máquinas

M.Rajoy

Al lo largo de este capítulo se detallan los pasos necesarios para la correcta configuración del sistema. En cada apartado se especifican los pasos a seguir para la configuración e instalación de los distintos servidores y software necesarios.

3.1. Virtualización del sistema

Debido a la gran cantidad de componentes necesarios para el desarrollo del proyecto, la necesidad de realizar el montaje y desmontaje de forma manual de estos y la dificultad en el acceso y configuración de cada uno de los nodos que lo componen decidimos realizar una virtualización del sistema para minimizar los problemas antes descritos. Así, al disponer de un entorno virtual, el cual, replica el clúster real, se minimiza el tiempo necesario para la configuración de los distintos servicios y servidores del sistema operativo y sirve, a su vez, como banco de pruebas para el desarrollo de software.

Para ello, hemos utilizamos *VMware workstation 12* como plataforma de software de vir-

tualización y una **imagen** de Raspbian basada en *Debian Stretch*, la cual se encuentra disponible en la página oficial de Raspberry Pi¹. Aunque el sistema del clúster parte de una versión diferente de Raspbian, el sistema de carpetas y, sobre todo, la instalación de **SIMCAN**, es similar al del entorno real.

En el repositorio en *Github*² existe una réplica del sistema de carpetas tanto para el servidor como para los nodos esclavo. Cada una de las carpetas y ficheros modificados en el sistema durante la configuración del sistema quedan reflejados en el repositorio, disponiendo así de un listado en forma de árbol de todas las configuraciones necesarias para el correcto funcionamiento de este.

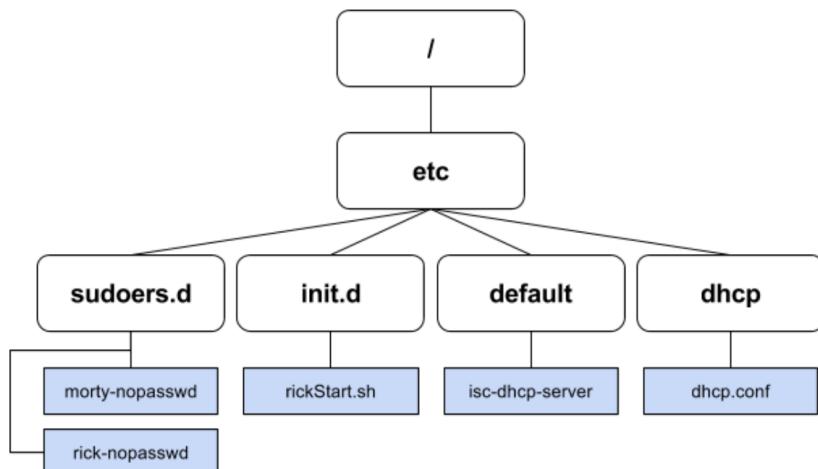


Figura 3.1: Sistema de archivos del repositorio

3.2. Sistema centralizado

Hemos decidido realizar nuestro sistema con una arquitectura de cluster computing centralizado para obtener más rendimiento y paralelizar aplicaciones, de este modo aprovechamos todos los recursos del cluster en su ejecución.

¹<http://downloads.raspberrypi.org/raspbian/images/>

²<https://github.com/migurome/RickAndMortys>

Disponemos de un nodo maestro que contiene el front-end y el back-end y realiza el envío de las peticiones a los esclavos, es escalable pudiendo incorporar nuevos nodos a la red, además ofrece una mayor seguridad sobre los sistemas descentralizados ya que todo el procesamiento es controlado a través de una localización central.

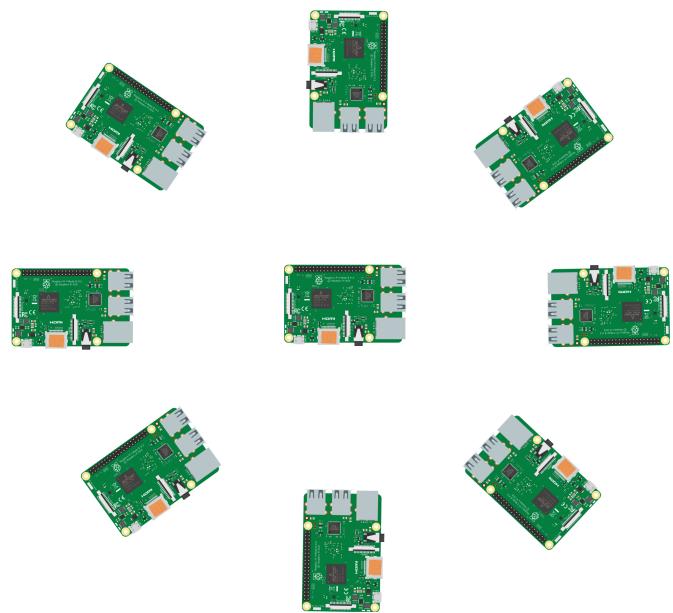


Figura 3.2: Sistema centralizado

3.3. Arquitectura del Clúster

El clúster está dividido en dos partes diferenciadas. Por un lado, el *front-end*, servidor, nodo del clúster que tiene instalado el software de **SIMCAN**, además de ofrecer los servicios de **NFS**, **DHCP** y **SSH** entre otros nodos como muestra la figura 3.3. Además, el punto de comunicación con el exterior, por ello dispone de una tarjeta de red adicional.

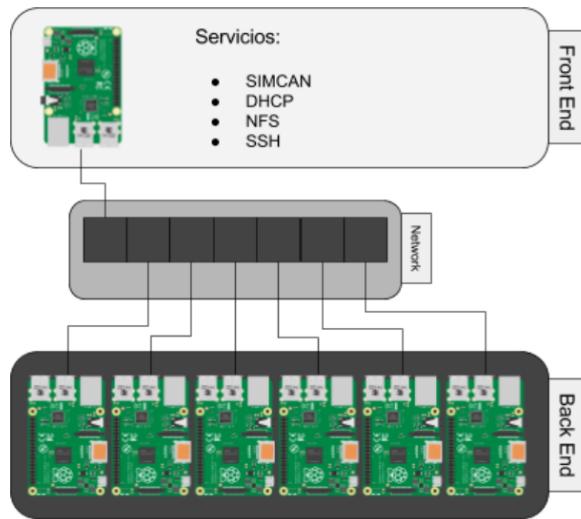


Figura 3.3: Esquema del sistema

Por otro lado, en el *back-end*, disponemos de varios nodos esclavo que disponen únicamente del sistema Raspbian Jessie, así como de bibliotecas necesarias para realizar las operaciones de cómputo.

3.4. Configuración de la red

La configuración de la red se realiza mediante DHCP, en este caso utilizaremos.....

La red en la que trabajamos es la $169.254.12.0/24$. El front end tiene asignada la dirección $169.254.12.1/24$, y ejerce de servidor sobre el resto de nodos, encargándose del reparto de direcciones, como se muestra en la figura 3.4.

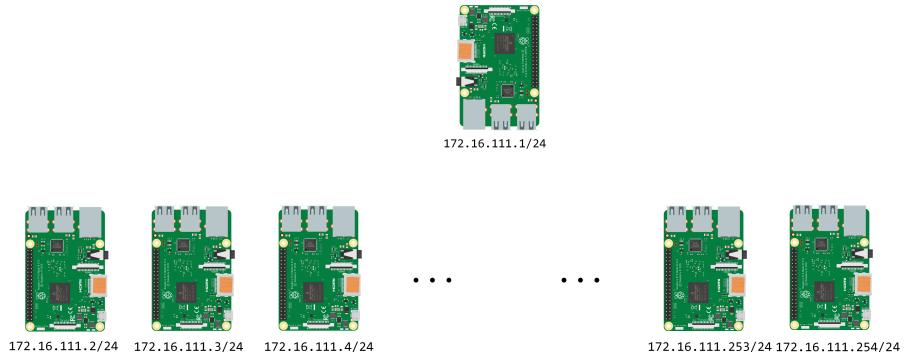


Figura 3.4: Servidor DHCP

Gracias al uso del protocolo **DHCP** evitamos tener que asignar manualmente direcciones a todos los nodos, sin embargo, es más útil en la práctica que cada uno de los nodos disponga de la misma dirección el máximo tiempo posible.

¿Por qué??

Para ello, en el fichero `/etc/dhcp/dhcpd.conf`, hemos establecido el parámetro **default-lease-time** que determina el tiempo de concesión de que el servidor asigna a cada nodo en su máximo valor. El tiempo por defecto (7776000 expresado en segundos) será de noventa días.

```
...
default-lease-time 7776000;
...
```

Los servicios ofrecidos por el nodo maestro de cara al *front-end* están disponibles a través de su segunda tarjeta de red, configurada para obtener una dirección IP desde un **ISP**.

3.4.1. Configuración de DHCP

METER ALGO DE TEXTO

Instalación de paquetes en el servidor

```
1 sudo apt-get update
```

```
2 sudo apt-get install isc-dhcp-server
```

Editar fichero **/etc/default/isc-dhcp-server**

```
3 sudo nano /etc/default/isc-dhcp-server
4 INTERFACES = "eth0"
```

Editar fichero **/etc/network/interfaces**

```
5 nano /etc/network/interfaces
6 source-directory /etc/network/interfaces.d

auto eth0
iface lo inet loopback
iface eth0 inet static

address 169.254.12.1
netmask 255.255.255.0
network 169.254.12.0

allow-hotplug wlan0
iface wlan0 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

allow-hotplug wlan1
iface wlan1 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Editar fichero de configuración **/etc/dhcp/dhcpd.conf**

Incluir la siguiente configuración al final del fichero:

```
subnet 169.254.12.0 netmask 255.255.255.0 {
    range 169.254.12.2 169.254.12.254;
}

host morty{
    hardware ethernet b8:27:eb:b3:36:02;
}
```

Activación de la red eth0 y reinicio

```
7 sudo ifconfig eth0 up  
8 sudo reboot now
```

Arranque del servicio DHCP

```
9 sudo /usr/sbin/dhcpd
```

Comprobación del correcto funcionamiento del sistema

```
10 ps -ef | grep dhcpcd
```

Opcionalmente se pueden mostrar las máquinas conectadas al servicio mediante la orden

```
11 cat /var/lib/dhcp/dhcp.leases
```

3.4.2. Hostname

A fin de diferenciar los distintos nodos de la red, a través de su *shell*, es necesario modificar el *hostname* de cada uno de ellos, para ello es necesario modificar los ficheros **/home/hostname** y **/home/hosts** en cada nodo.

Fichero **/home/hosts**

```
127.0.0.1 localhost  
::1 localhost ip6-localhost ip6-loopback  
ff02 ::1 ip6-allnodes  
ff02 ::1 ip6-allrouters  
  
127.0.0.1 morty_x
```

Fichero **/home/hostname**

```
morty_x
```

Es necesario reiniciar la máquina para que se apliquen correctamente los cambios.

3.4.3. Network File System (NFS)

Como se destacaba anteriormente, el front-end es el único que dispone de una versión de **SIMCAN** instalada, con esto se evitan problemas de versiones y es mas sencillo realizar actualizaciones, esta configuración ofrece la posibilidad de que la labor del los nodos esclavo sea únicamente la de realizar el procesamiento de datos. Mediante **NFS**, el nodo servidor comparte su carpeta `/home/` durante el arranque del sistema. De esta forma, el resto de nodos esclavo realizan el montaje de este directorio compartido en red en su propio directorio `/home/`, creando así un único punto de acceso compartido en red del que se pueden extraer los ejecutables sin la necesidad de disponer de **SIMCAN** instalado. El front-end se encarga de realizar la compilación los ficheros `.ned` y pone a disposición del resto de nodos los ejecutables.

Es necesario que todos los nodos de la red tengan un mismo usuario común para conseguir una correcta sincronización, de igual manera hay que mantener un estricto control de los permisos de cada uno de los nodos esclavo tanto a nivel interno como de cara al servidor.

3.4.4. Instalación de NFS

PARRAFO DE INTRODUCCION.....

Instalar paquetes en el servidor

```
1 sudo apt-get update  
2 sudo apt-get install nfs-kernel-server
```

Editar fichero de configuración `/etc(exports` en servidor

```
Ruta de carpeta / Rango de direcciones destino / Permisos  
/home/pi 169.254.12.0/24(rw, sync, no_subtree_check, no_root_squash)
```

Instalación de paquetes en el cliente

```
3 sudo apt-get update
```

```
4 sudo apt-get install build-essential gcc g++ bison flex perl tcl-dev tk-dev libxml2-dev zlib1g-dev default-jre doxygen graphviz libwebkitgtk-1.0-0 openmpi-bin libopenmpi-dev libpcap-dev
```

Ejecutar cambios realizados en el servidor

```
5 sudo exportfs -a  
6 sudo /etc/init.d/rpcbind restart  
7 sudo /etc/init.d/nfs-kernel-server restart
```

Finalmente, ejecutamos el siguiente comando en el cliente para realizar el montaje del directorio:

```
8 sudo mount -t nfs 169.254.12.1:/home/pi /home/pi
```

3.4.5. Creación y ejecución del servidor NFS como un *daemon* del sistema

Para evitar tener que realizar el arranque del servidor de forma manual es recomendable crear un *daemon* e incluirlo en el directorio **/etc/init.d** para que se ejecute con el arranque del sistema de forma automática tanto en el nodo servidor como en los nodos cliente.

Contenido del script, sevidor:

```
#!/bin/bash  
### BEGIN INIT INFO  
# Provides: M. Romero && D. Quinones  
# Required-start: $syslog  
# Required-stop: $syslog  
# Default-Start: 2 3 4 5  
# Default-Stop: 0 1 6  
# Short-Description: Inicialización de servicios nfs  
# Description:  
### END INIT INFO  
sudo exportfs -a  
sudo /etc/init.d/rpcbind restart  
sudo /etc/init.d/nfs-kernel-server restart
```

Contenido del script cliente, el comando *sleep 10* simplifica errores de *timming* durante el montaje del servicio *NFS*:

```
#!/bin/bash
### BEGIN INIT INFO
# Provides: M.Romero
# Required-start: $syslog
# Required-stop: $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Arranque y montaje de NFS
# Description:
### END INIT INFO
sleep 10
sudo mount -t nfs 169.254.12.1:/home/pi /home/pi
```

El siguiente cuadro muestra los pasos a seguir para establecer el script como un *daemon* del sistema, se realizan de forma automática la creación de enlaces simbólicos y situación de los ficheros en **/etc/rcrunlevel.d** correspondiente:

```
1 chmod +x nombre_del_script.sh
2 cp nombre_del_script.sh /etc/init.d/
3 cd /etc/init.d
4 update-rc.d nombre_del_script.sh defaults
```

3.5. Instalación de OMNeT++, INET y SIMCAN

Antes de poder instalar el software **SIMCAN** es necesario realizar la instalación previa framework **OMNeT++** en su versión 4.6. Además de la suite **INET**, que implementa modelos de código abierto **OMNeT++** para redes cableadas, inalámbricas y móviles.

Debido a la baja potencia de la Raspberry, ésta no es capaz de lanzar la aplicación de forma gráfica. Esto supone un problema a la hora de realizar la instalación del software. Es por esto que todas las instalaciones han de realizarse a través del terminal. Esto afecta principalmente a la instalación de **INET**, ya que las principales guías de instalación disponibles en las webs oficiales parten siempre del entorno gráfico de **OMNeT++**.

Durante el desarrollo del proyecto se han desarrollado una serie de guías para la instalación y configuración que se desglosarán en el siguiente apartado.

3.5.1. Instalación de OMNeT++

FRASE DE INTRO

Descargar los *tar.gz* de OMNeT++ 4.6, INET, simcan.tar. Esta última (simcan) incluye las bibliotecas que se necesitan para la compilación. Copiar los archivos *.tar* de OMNeT++ e INET en **/pi** y descomprimir. Desde el directorio **/pi** ejecuta los siguientes comandos:

```
1 sudo apt-get update
2 sudo apt-get install build-essential gcc g++ bison flex perl tcl-dev tk-
   dev libxml2-dev zlib1g-dev default-jre doxygen graphviz libwebkitgtk-1.0-0
   openmpi-bin libopenmpi-dev libpcap-dev
3 sudo apt-get install gnome-color-chooser
4 cd omnetpp-4.6
5 . setenv
6 ./configure
7 make
# Opcionalmente podemos utilizar el comando make -j 'numero de cores' para
# paralelizar el compilado del proyecto
```

3.5.2. Instalación de INET

FRASE DE INTRO

Crear un directorio nuevo en **/omnetpp-4.6** llamado *projects*, copiar en el directorio **INET** descomprimido y ejecutar:

```
8 sudo apt-get install libavcodec-dev libavformat-dev
9 make makefiles
10 make
# Opcionalmente podemos utilizar el comando make -j 'numero de cores' para
# paralelizar el compilado del proyecto
```

3.5.3. Instalación de SIMCAN

FRASE DE INTRO

Copia el directorio de **simcan** a **/projects** y ejecuta los siguientes comandos:

```
11 export omnetpp_root=$HOME/morty/omnnetpp-4.6
12 export INET_HOME=$omnetpp_root/projects/inet
13 export SIMCAN_HOME=$omnetpp_root/projects/simcan
14 export LD_LIBRARY_PATH=$omnetpp_root/lib:$LD_LIBRARY_PATH
15 export PATH=$omnetpp_root/bin:$PATH
16 make makefiles
17 make
# Opcionalmente podemos utilizar el comando make -j 'numero de cores' para
# paralelizar el compilado del proyecto
Enjoy!
```

3.5.4. Problemas

1. No se encuentra el fichero **TCPCommand_m.h**

```
cp /home/pi/OMNeT++pp-4.6/projects/INET/src/transport/contract/
TCPCommand_m.h /home/pi/OMNeT++pp-4.6/projects/simcan/src/Messages/
TCPCommand_m.h
```

2. Error al ejecutar **run_simcan**

```
chmod +x /simcan/src/run_sincam
```

3.6. Configuraciones derivadas de la arquitectura

Como se ha explicado anteriormente, la arquitectura elegida distribuye la potencia a todos los nodos del clúster desde una misma fuente de alimentación. Debido a la menor cantidad de recursos y servicios que han de ofrecer los nodos esclavo, éstos tienen una carga del sistema

ligeramente más rápida que la del nodo maestro. Por ello, se pueden producir problemas de sincronización de servicios, más concretamente, en el montaje de sistemas de ficheros compartidos en red por **NFS**, el cual es crítico para el funcionamiento general del sistema. Para solucionar este problema se han realizado modificaciones a fin de conseguir acelerar la carga del nodo maestro y ralentizar la del resto.

3.6.1. Modificación del GRUB

Una de las soluciones más sencillas para resolver el problema de sincronización consiste en aumentar el tiempo por defecto del *grub* de los nodos esclavo ya que la carga del sistema se produce cuando éste termina. Para ello únicamente hay que modificar la opción **GRUBTIMEOUT** en el fichero **/etc/default/grub**. En las pruebas realizadas durante la virtualización del sistema se comprobó que estableciendo un retardo de veinticinco segundos era suficiente para que el nodo maestro realizase la carga completa del sistema. Sin embargo, queda por comprobar que en el entorno real esto se sigue produciendo.

Contenido del fichero **/etc/default/grub** de un nodo esclavo:

```
GRUB_DEFAULT=0
GRUB_TIMEOUT=25
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash plymouth.ignore-serial-consoles"
GRUB_CMDLINE_LINUX=""
```

3.6.2. Habilitar arranque automático de un usuario

Para modificar el usuario de arranque por defecto es necesario realizar los siguientes cambios en el fichero **/etc/lightdm/lightdm.conf**

```
autologin-user = nombre_de_usuario
```

3.6.3. Forzar el arranque sin HDMI

En ocasiones *Raspbian Jessie* no realiza la carga del sistema operativo si el conector **HDMI** no está acoplado en la Raspberry, para forzar el arranque del sistema es necesario modificar el archivo **/boot/config.txt**

```
#Arranque sin HDMI  
hdmi_force_hotplug = 1
```

3.7. Seguridad

3.8. Eliminar usuarios y permisos

Capítulo 4

Diseño de Prototipos

*La cosa está muy mal... estoyriendo los huevos
con saliva*

Gregorio Esteban Sánchez

En este capítulo se muestra el diseño de los distintos modelos que se han generado para el proyecto. Se dispone, para cada uno, una vista en 3D, creada con *OpenScad*, que representa la disposición de los distintos componentes y la dirección de las corrientes de aire generadas por los ventiladores, finalmente, se incluyen capturas del resultado de cada uno de ellos.

4.1. Características del modelado

Los componentes representados en el modelo en 3D son: el switch de ocho puertos ([Azul](#)), un clúster de Raspberrys formado por siete nodos ([Verde](#)), la base de enchufes ([Rojo](#)) y los ventiladores ([Gris](#)), no se incluyen los cables y conectores conectados a la base y a cada Raspberry.

En cuanto a la disposición de los ventiladores, para cada uno de los modelos, se han realizado diferentes configuraciones, todas ellas destinadas a obtener la corriente de aire más eficiente y la mejor tasa de temperatura cuando los nodos están trabajando a máximo rendimiento.

Los resultados de estas se muestran en el capítulo [5](#), en donde se verá el comportamiento

de cada una con una serie de pruebas destinadas a medir la temperatura ante distintos escenarios.

4.2. Modelo 1

En este modelo la distribución de aire se hace con los ventiladores opuestos entre sí, creando una corriente de aire alrededor del rack de Raspberrys que se encuentra frente a ellos.

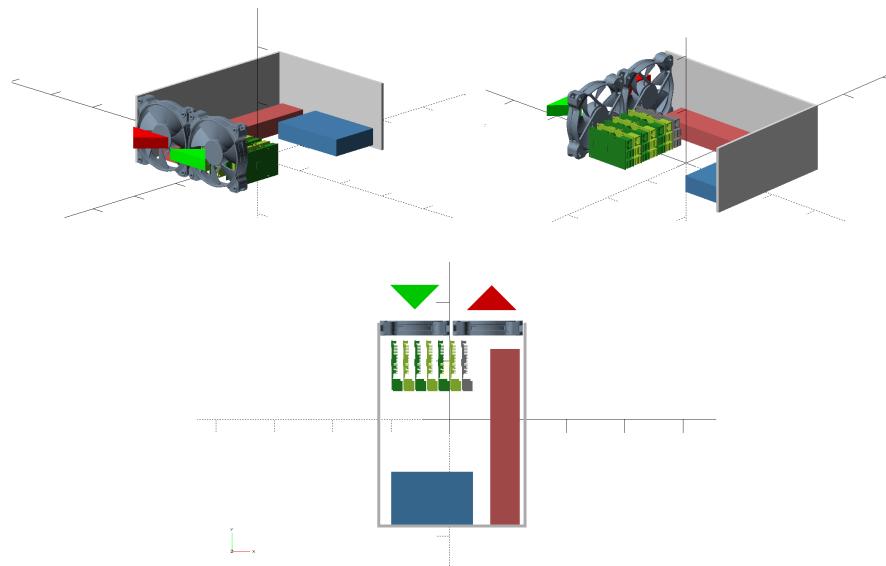


Figura 4.1: Modelo 1 3D

La figura 4.1 muestra la vista trasera, delantera y alzada del modelo. La prioridad es obtener una buena distribución de aire, no se ha tenido en cuenta la facilidad para acceder a los componentes. Para poder acceder a las tarjetas **SD** de cada Raspberry es necesario mover todo el rack, lo que hace que sea complicado, una solución alternativa sería ofrecer la opción de desacoplar uno de los ventiladores para facilitar el acceso.

4.2.1. Resultado

Este modelo tiene un tamaño de 122x123x35 cm y en la captura no se han incluído los cables rj45 que conectan el rack de Raspberrys con el switch. Podemos observar que los componentes disponen de un espacio muy limitado y que la poca flexibilidad de los cables dificulta el cierre del contenedor.



Figura 4.2: Resultado Modelo 1

Las figura 4.2 muestra el resultado del diseño descrito en el apartado anterior, se puede observar la distribución de los componentes dentro del contenedor y el cierre del que dispone.

4.3. Modelo 2

Para resolver los problemas de accesibilidad al rack este modelo dispone de una apertura lateral que expone las tarjetas **SD** de cada Raspberry, con lo que no es necesario abrir el contenedor para extraer cada tarjeta. Además el contenedor está dividido en dos piezas, una que se acopla sobre la otra para formar un cubo, esto facilita la apertura y acceso al mismo.

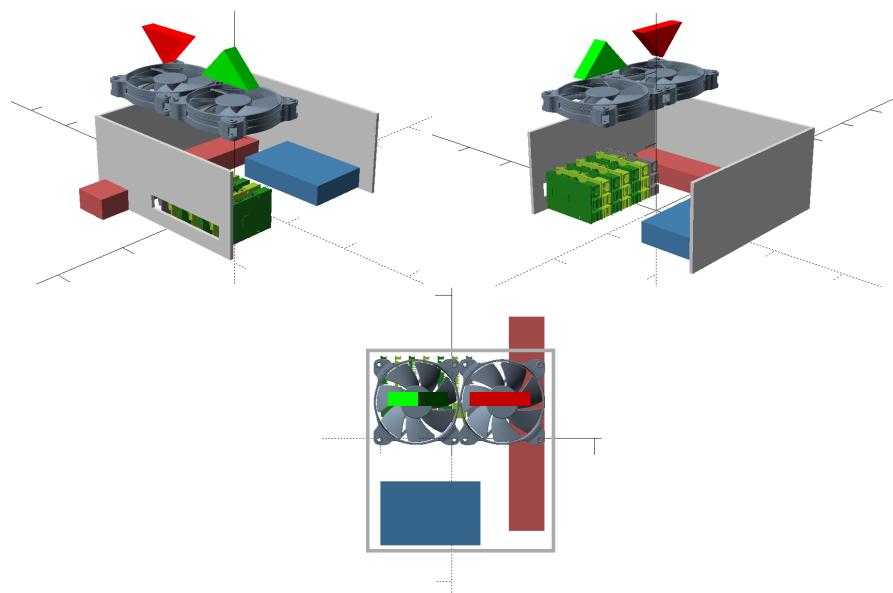


Figura 4.3: Modelo 2 3D

Como se muestra en la figura 4.3 la ventilación se realiza desde la parte superior, creando, como en el caso anterior, una corriente de aire debida al emplazamiento de cada ventilador. Se puede observar, en la vista frontal, la apertura centrada sobre el rack de Raspberrys. Este modelo es más compacto que el anterior, parte de la base de enchufes queda en el exterior. Sin embargo, existe un mayor espacio entre los ventiladores y el rack. La disposición de los cables de alimentación del rack es mejor que la del modelo 1, junto con el acceso lateral, prácticamente no es necesario manipular ninguno de los componentes.

4.3.1. Resultado

Este modelo tiene un tamaño de 122x123x35 cm, en la figura 7.2 se puede observar el sistema de acople en dos piezas que abren el contenedor. Como se ha comentado anteriormente, parte de la base de enchufes queda en el exterior, esto supone una mejora ya que permite el encendido y apagado del clúster sin que sea necesario abrir todo el contenedor para ello.



Figura 4.4: Resultado Modelo 2

Como resultado de todas estas mejoras tenemos un modelo que es mucho mas manejable que el anterior, de tamaño más reducido y con mejoras estructurales que facilitan la manipulación.

4.4. Modelo 2b

El modelo 2b presenta unas modificaciones estructurales internas, tras hacer un estudio previo sobre corrientes de aire decidimos realizar algunas modificaciones sobre éste para conseguir mejorar la eficiencia del flujo de aire dentro del contenedor.

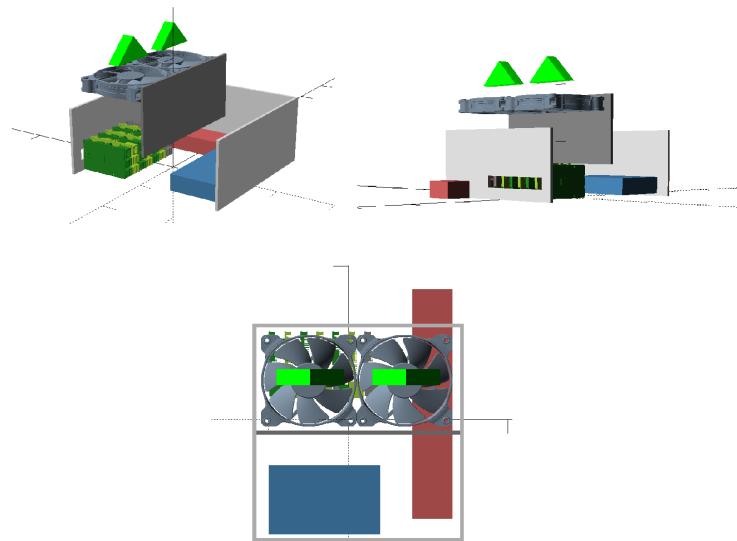


Figura 4.5: Modelo 2b 3D

Para esto, se incluye una nueva pared interior, distribuyendo el contenedor en *dos cámaras*, una que contiene el rack de Raspberrys y otra con el switch. La corriente de aire se concentra en la primera cámara, con esto generamos un flujo que entra en el contenedor a través de la entrada lateral, en la que están expuestas las Raspberrys, pasando por cada uno de los procesadores y saliendo por la parte superior en la que ambos ventiladores expulsan el aire. La figura 4.5, similar a la del modelo 2, muestra la separación en cámaras y distribución de los ventiladores. Como se puede comprobar en el capítulo 5 esta modificación ofrece unos resultados completamente distintos en cuanto a las temperaturas del contenedor.

4.5. Modelo 3

Hemos diseñado el modelo 3 utilizando la arquitectura de *túnel de viento*, éste consiste en suministrar una corriente de aire de entrada sobre las partes que más tienden a calentarse, en este caso, el rack de Raspberrys, aplicando una corriente de succión con un ventilador de salida, creando un flujo de aire horizontal.

4.5.1. Modelado 3D

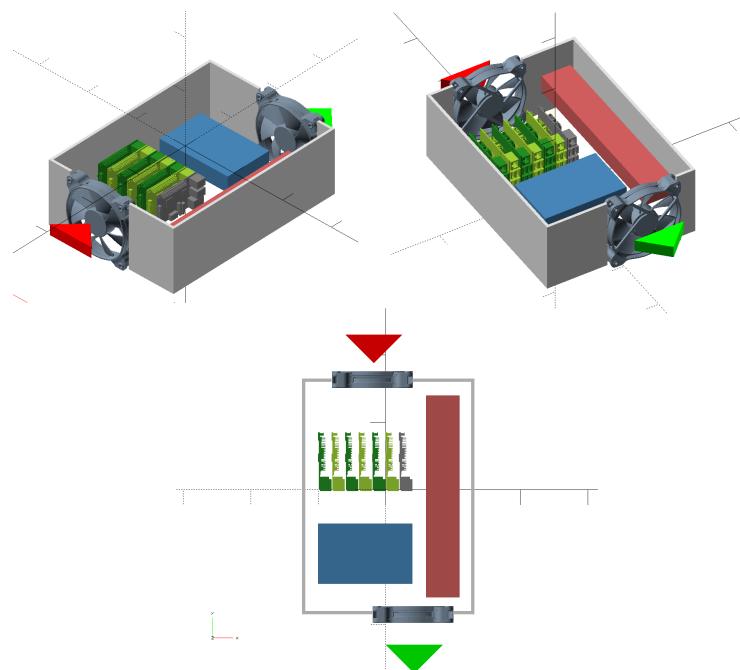


Figura 4.6: Modelo 3 3D

Debido a las altas temperaturas presentadas por el modelo 1 decidimos aplicar una corriente de aire directa sobre las Raspberrys y sacarla del sistema para evitar bucles de aire calientes a través del segundo ventilador que está situado al lado del switch como se muestra en la figura 4.6

Tras una serie de pruebas y mediciones de temperatura con la disposición de los ventiladores,

entrada-salida, entrada-entrada y salida-salida, llegamos a la conclusión que la más eficiente era la primera de ellas.

4.5.2. Resultado

Tras el desarrollo de los modelos anteriores, hemos creado un híbrido entre el modelo 1 y 2B, del primero aprovechamos su base y el ventilador del que ya disponíamos, del segundo, hemos decidido que en vez de aplicar la corriente de succión en la parte de arriba, le otorgaremos un flujo de aire que recorre todo el contenedor a través del túnel de viento ya mencionado. Mientras los anteriores modelos se centraban en las Raspberrys, este proporciona una corriente tanto al Switch como al rack.



Figura 4.7: Resultado Modelo 3

El T-800 de la figura 4.7 no está incluido en el pack, presentó demasiadas complicaciones a nivel de diseño y entraba continuamente en conflicto debido a las tres leyes de la robótica.

Capítulo 5

Tests de Diseño

Tendrá todo el dinero del mundo, pero hay algo que nunca podrá comprar... un dinosaurio.

Homer J. Simpson

El objetivo de este capítulo es el de mostrar el resultado de las pruebas de temperaturas realizadas en los diferentes modelos diseñados en el capítulo 4.

5.1. Experimental Settings

Se ha sometido a estos a un total de tres pruebas, todas ellas con las mismas características para cada modelo, variando en el número de nodos que hay trabajando de forma simultánea en cada caso. De esta manera, según los resultados obtenidos y las características de diseño generadas en el capítulo 4 determinamos qué modelo es el más idóneo para la elección final. Además de las pruebas generales, se han realizado una serie de pruebas específicas para cada modelo, el objetivo de estas, ha sido el de determinar la mejor configuración en cuanto a la disposición de los ventiladores dentro del contenedor. En este capítulo, únicamente se muestran las pruebas realizadas a los modelos finales.

Para el desarrollo de cada prueba se ha generado un script que obtiene información de la temperatura del procesador ofrecida directamente por el sistema operativo. El siguiente

cuadro muestra el contenido del script:

```
#!/bin/bash
while test 0 -eq 0
do
    sleep $1
    temp=$( /opt/vc/bin/vcgencmd measure_temp | cut -c 6-9)
    name=$( cat /home/hostname )
    echo "$name $temp"
done
exit 0
```

5.2. Prueba 1

5.2.1. Escenario

En esta prueba únicamente hay una Raspberry Pi trabajando con cuatro cores, el tiempo total de cada prueba es de tres horas, el periodo de muestreo es de diez segundos y en las gráficas se muestra el tiempo total expresado en segundos. La temperatura ambiente oscila entre los catorce y dieciocho grados. Ninguna de las raspberrys utilizadas dispone de un dissipador instalado sobre el procesador.

5.2.2. Resultados

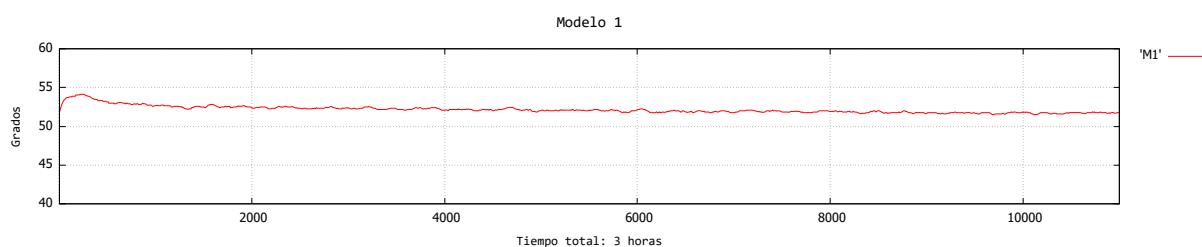


Figura 5.1: Modelo 1



Figura 5.2: Modelo 2

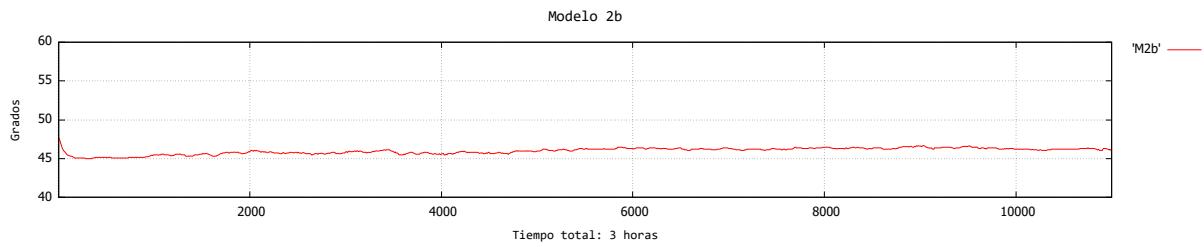


Figura 5.3: Modelo 2b

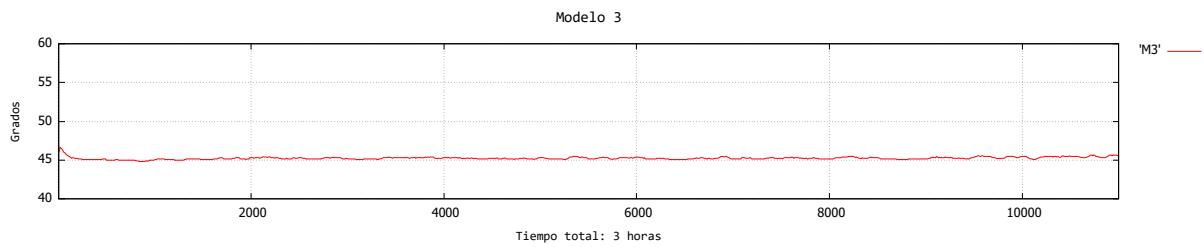


Figura 5.4: Modelo 3

5.2.3. Conclusiones

Se puede comprobar que tanto el modelo 2b como el modelo 3 son los que obtienen unos mejores resultados, en todo caso, en todos ellos, el sistema se mantiene estable durante toda la prueba, con muy poca variación entre sus temperaturas máxima y mínima. El modelo 2 registra unos valores cercanos a los sesenta grados, con lo que es sin duda el peor de los tres.

5.3. Prueba 2

5.3.1. Escenario

En esta prueba hay tres Raspberrys trabajando de forma simultánea, cada una de ellas mantiene sus cuatro cores trabajando a máximo rendimiento, el tiempo total de las pruebas es nuevamente de tres horas, igual que en la prueba anterior el periodo de muestreo es de diez segundos y en cada gráfica muestra el tiempo total expresado en segundos. La temperatura ambiente, en este caso, oscila entre los quince y dieciocho grados. Una de las Raspberrys dispone de un disipador de calor, en las gráficas, esta se identifica con la letra D al lado de su nombre.

5.3.2. Resultados

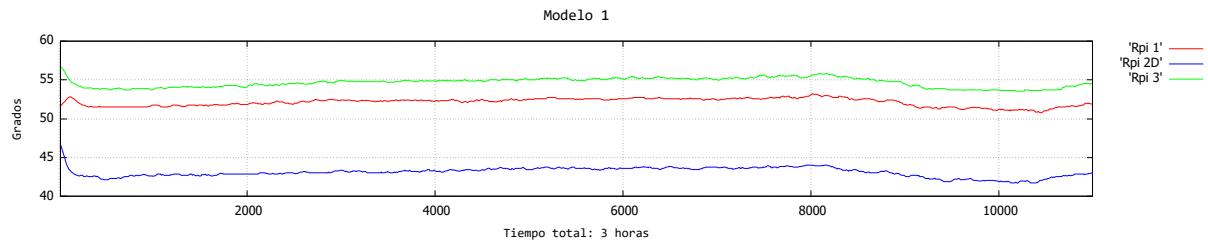


Figura 5.5: Modelo 1

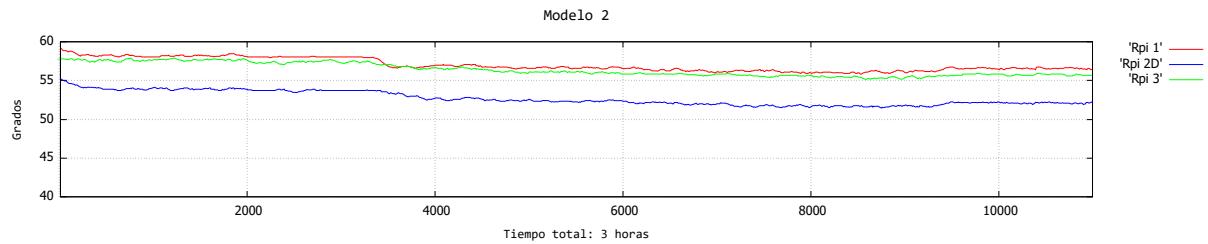


Figura 5.6: Modelo 2

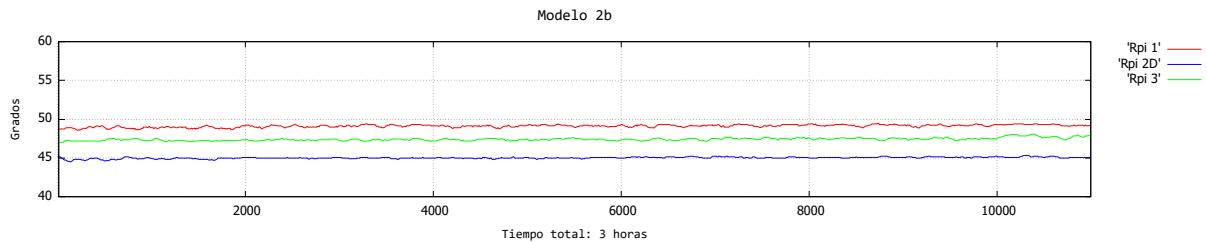


Figura 5.7: Modelo 2b

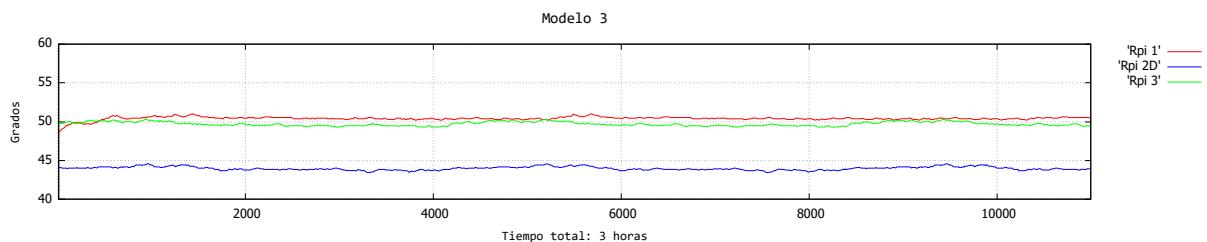


Figura 5.8: Modelo 3

5.3.3. Conclusiones

Hemos comprobado que una buena corriente de aire hace que aquellos nodos que disponen de un disipador de calor instalado presenten unas temperaturas notablemente inferiores al resto. Sin embargo, como pasa en el modelo 2, figura 5.6, si la corriente no es eficiente, dicha mejora se ve severamente afectada.

De nuevo, los modelos 2b y 3, ofrecen unos resultados mejores, además se mantienen más estables durante toda la prueba. El modelo 1, figura 5.5, es en el que más variación de temperatura se observa entre Raspberrys con y sin disipadores. Nuevamente el modelo 2 tiene unas temperaturas superiores, cercanas a los sesenta grados. Vemos además que el hecho de incluir más nodos de cómputo no afecta a las temperaturas medias de cada uno, ofreciendo resultados similares a los de la primera prueba.

5.4. Prueba 3

5.4.1. Escenario

Finalmente en esta última prueba hay seis Raspberrys trabajando de forma simultánea, como en los casos anteriores, cada una de ellas mantiene sus cuatro cores trabajando en paralelo, el tiempo total de las pruebas es nuevamente de tres horas y el muestreo se realiza cada diez segundos. La temperatura ambiente oscila entre los dieciséis y diecisiete grados. Para esta prueba existen dos Raspberrys que disponen de disipadores de calor, identificadas nuevamente con la letra D en la gráfica.

5.4.2. Resultados

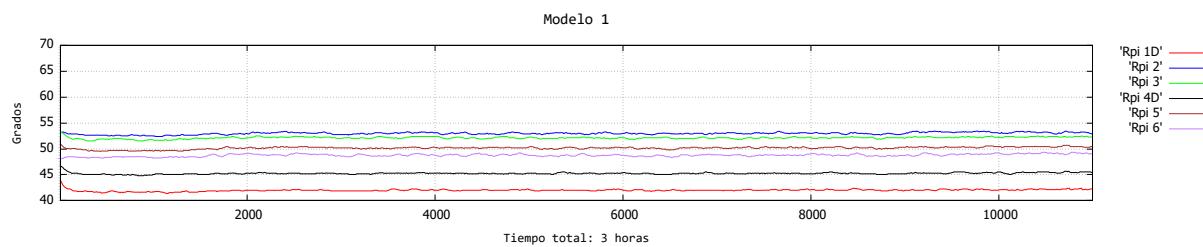


Figura 5.9: Modelo 1

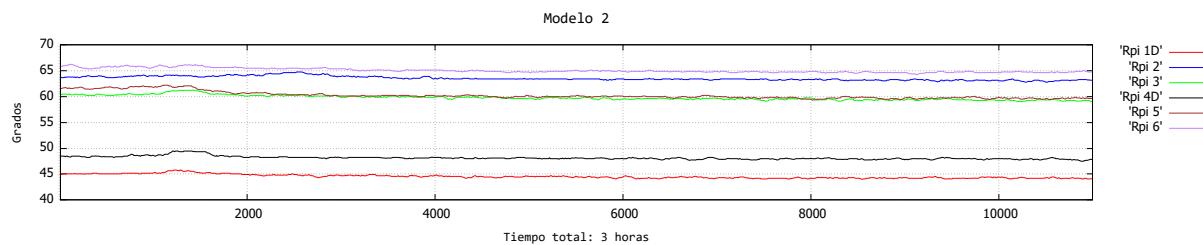


Figura 5.10: Modelo 2

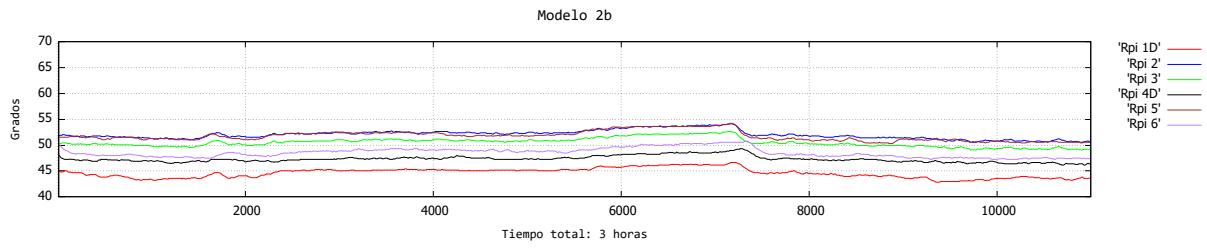


Figura 5.11: Modelo 2b

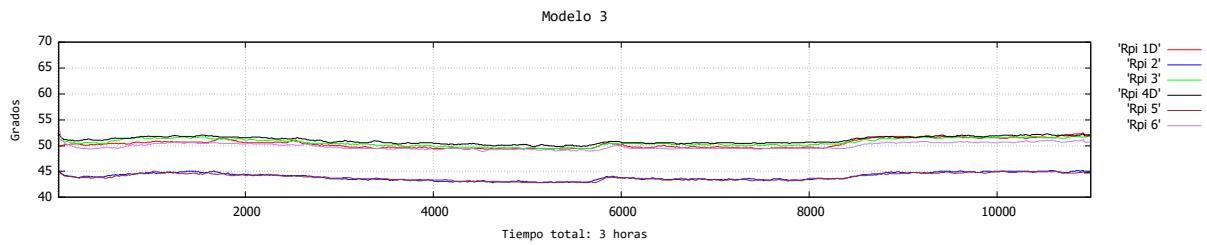


Figura 5.12: Modelo 3

5.4.3. Conclusiones

En esta prueba hemos detectado que definitivamente el modelo 2 no cumple con las expectativas necesarias para el correcto funcionamiento del clúster, en él, las diferencias entre nodos con y sin disipador son muy notables y aquellas que no disponen de esta mejora arrojan unas temperaturas muy próximas a los ochenta grados, temperatura crítica de parada, por lo que podemos descartar este modelo para el desarrollo final.

Los otros tres modelos ofrecen unos resultados semejantes a las anteriores pruebas, particularmente el modelo 1 sigue mostrando gran diferencia entre los nodos con disipador. El modelo 2b, aunque ofrece algunas variaciones más notables durante la prueba, sigue manteniendo una relación parecida entre nodos, independientemente de que incluya o no disipadores.

El modelo 3 presenta los mejores resultados, manteniendo temperaturas bajas cercanas a los cincuenta grados y una notable diferencia entre las Raspberrys que disponen de disipador, las cuales se mantienen en cuarenta y cinco grados.

5.5. Prueba 4

5.5.1. Escenario

El objetivo de esta última prueba es comprobar el comportamiento del sistema durante un periodo de tiempo largo. Se realizó sobre el modelo 1 durante venticuatro horas de forma ininterrumpida. El escenario es similar al de la prueba 3, con seis Raspberrys trabajando en paralelo, sin embargo, este muestreo se realiza cada sesenta segundos. No podemos certificar la temperatura ambiente para toda la prueba, pero calculamos que osciló entre los quince y dieciocho grados.

5.5.2. Resultados

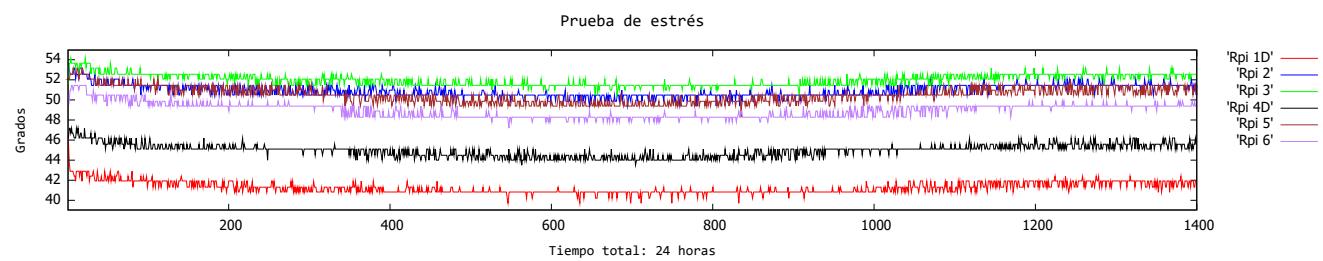


Figura 5.13: Resultados

5.5.3. Conclusiones

Prueba de estrés

5.6. Conclusiones generales

Durante el desarrollo de los test hemos podido apreciar algunos de los diferentes factores más influyentes sobre el sistema, a continuación, destacamos los que han sido más notorios:

1. Las *corrientes de aire* tienen una gran afección, merece la pena dedicar parte del trabajo a buscar una eficiencia en la conducción de flujos de aire dentro del contenedor, una buena distribución de la entrada y salida de aire nos permite reducir el número de ventiladores y establecer un punto de partida para la disposición del resto de elementos.
2. Al contar con un gran número de cables, también es necesario combinar el punto anterior con la *accesibilidad general a los dispositivos*, de esta manera, exponiendo las tarjetas SD conseguimos que no sea necesario abrir el contenedor para acceder a cada uno de los nodos.
3. La *temperatura ambiente* es otro de los factores influyentes en el comportamiento general, aunque menos notorio, se pueden apreciar alteraciones en todos los dispositivos sujetos a este efecto. Sería preciso que ésta se mantuviera lo más estable posible.
4. En cuanto a la *robustez*, los resultados ofrecidos por la prueba de estrés, figura 7.5, muestran que el comportamiento ha sido similar al del resto de pruebas, independientemente del número de horas, por lo que podemos concluir que el sistema es fiable y poco propenso a las caídas repentinas por periodos largos de trabajo.
5. En las diferentes pruebas realizadas hemos observado que *no existe una correlación entre tasa de trabajo completado y temperatura*. Aún así, con el fin de prolongar la vida útil de cada nodo, siempre es mejor mantener un rango de temperaturas bajo.
6. El uso de *disipadores* es muy conveniente, son elementos de precio reducido, y con una distribución de aire adecuada aumentan tanto la vida útil del clúster, como el

comportamiento de los nodos ante los cambios bruscos de temperatura ambiente. En el peor de los casos, todos los nodos que disponían de esta mejora ofrecieron unos resultados inferiores a los del resto.

Capítulo 6

Cliente Servidor en JAVA

'long long long' is too long for GCC

Some GCC programer

La introducción que creamos necesaria para este capítulo

6.1. Creación de un .jar

Para poder ejecutar los *.jar* correctamente debemos tener nuestra versión de *Java* en la **versión 1.8**, esto se configura de la siguiente manera:

```
1 sudo update-alternatives --config java
2 Elegiremos la opción /usr/lib/jvm/oracle-jdk8-jdk-i386/jre/bin/java
3 En caso de que no se encuentre:
   3.1 sudo apt-get upgrade
   3.2 sudo apt-get update
   3.3 sudo apt-get install default-jre
Dentro de eclipse:
4 Seleccionamos: Proyect - export - runnable JAR file
   4.2 launch configuration elegimos la clase que queremos hacer ejecutable
       (client.java o server.java)
   4.3 opciones seleccionadas:
       4.3.1 extract required libraries into generated jar
4.4 finish
```


Capítulo 7

Modelo Final

No te engañes Jimmy. Si una vaca tuviera la oportunidad, te comería a ti, y a los seres que tú mas quieres.

Troy McClure

Una introducción

7.1. Diseño Modelo

7.1.1. Diseño 3D

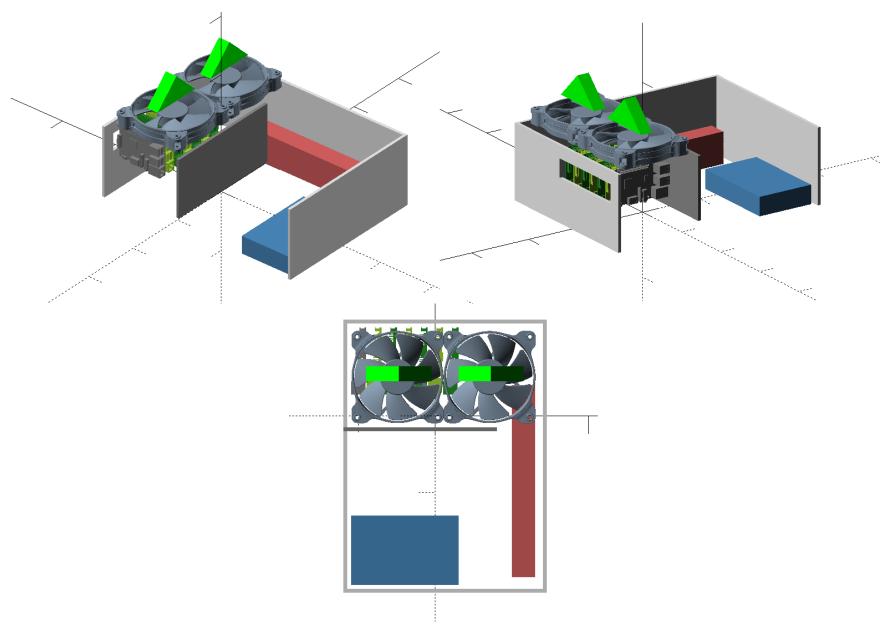


Figura 7.1: Modelo Final 3D

7.1.2. Resultado Modelo



Figura 7.2: Resultado Modelo Final

7.2. Pruebas Modelo

7.2.1. Resultados Pruebas

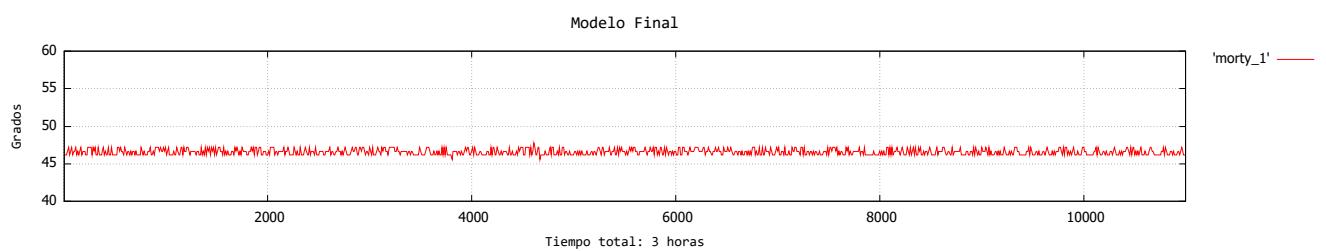


Figura 7.3: Resultados Prueba 1

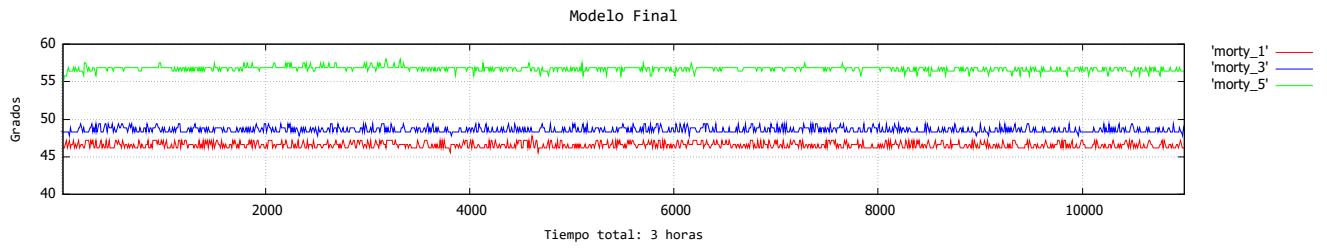


Figura 7.4: Resultados Prueba 2

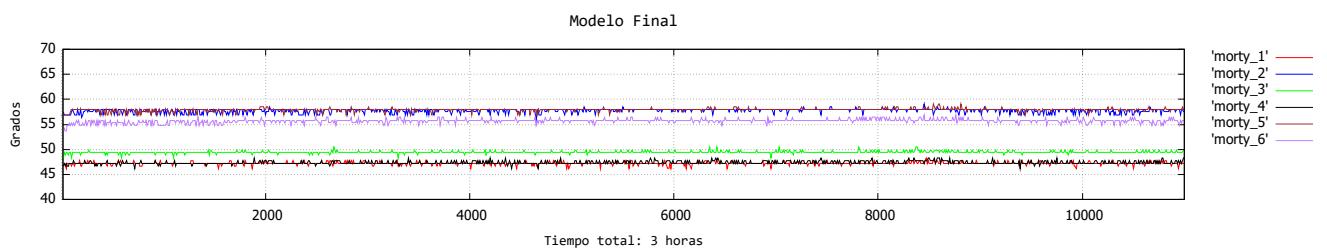


Figura 7.5: Resultados Prueba 3

1. Pruebas de temperatura con el modelo final
2. Pruebas de rendimiento con el modelo final
3. Código en Java

