



ESCUELA SUPERIOR DE INGENIERÍA

Grado en Ingeniería Informática

**Infraestructura de red de nodos  
cifradores/descifradores AES basada en ApSoC**

Curso 2019-2020

Jesús Rodríguez Heras

Puerto Real, 30 de Agosto de 2020





ESCUELA SUPERIOR DE INGENIERÍA

Grado en Ingeniería Informática

**Infraestructura de red de nodos  
cifradores/descifradores AES basada en ApSoC**

DEPARTAMENTO: Ingeniería en Automática, Electrónica, Arquitectura y Redes de Computadores.

DIRECTORA DEL PROYECTO: María Ángeles Cifredo Chacón.

CODIRECTORA DEL PROYECTO: María Mercedes Rodríguez García.

AUTOR DEL PROYECTO: Jesús Rodríguez Heras.

Puerto Real, 30 de Agosto de 2020

Fdo.: Jesús Rodríguez Heras



## Declaración personal de auditoría

Jesús Rodríguez Heras con DNI 32088516C, estudiante del título de Grado de Ingeniería Informática en la Escuela Superior de Ingeniería de la Universidad de Cádiz, como autor de este documento académico titulado “Infraestructura de red de nodos cifradores/descifradores AES basada en ApSoC” y presentado como Trabajo Final de Grado

### DECLARO QUE:

Es un trabajo original, que no copio ni utilizo parte de obra alguna sin mencionar de forma clara y precisa su origen tanto en el cuerpo del texto como en su bibliografía y que no empleo datos de terceros sin la debida autorización, de acuerdo con la legislación vigente. Asimismo, declaro que soy plenamente consciente de que no respetar esta obligación podrá implicar la aplicación de sanciones académicas, sin perjuicio de otras actuaciones que pudieran iniciarse.

En Puerto Real, a 30 de Agosto de 2020.

Fdo.: Jesús Rodríguez Heras



# **Agradecimientos**

Me gustaría mostrar mis agradecimientos a mis tutoras del proyecto, a mi familia, mi pareja y mis amigos que me han ayudado psicológicamente en todo el proceso motivándome día a día a seguir y terminar el último paso de mi carrera.





## **Resumen**

En este proyecto se ha trabajado en el diseño de una estructura de red que conecta un ordenador con unos nodos cifradores/descifradores basados en tecnología programable ApSoC. Para tal fin, los nodos incluyen un periférico hardware de cifrado/descifrado AES-128 implementado en la lógica programable ApSoC. Cada nodo de la red recibe información del nodo anterior, la descifra, agrega información vuelve a cifrar el conjunto para enviarlo al siguiente nodo. De este modo se recolecta información de todos los nodos de forma colectiva, enviándose al monitor central al finalizar.

Para la conexión de todos los dispositivos de red se han usado cables de red UTP de categoría 5E y un switch TP-Link TL-SG1024D. Se ha desarrollado un conjunto de scripts que automatizan el proceso al completo, encargándose por tanto de la recepción, cifrado/descifrado y envío del conjunto de información mediante el protocolo de comunicación SSH. Este proceso ha sido automatizado con el objetivo de conseguir una mayor independencia del agente humano por parte del sistema.

La red montada para el proceso de test y verificación del sistema, usa un total de tres nodos, sin embargo, el diseño permite aumentar el número de dispositivos en base a las necesidades y capacidades físicas de la red.



# **Palabras clave**

Red, Zynq, Conexión, AES, ApSoC.



# Contenido

<b>I</b>	<b>Contenido</b>	<b>21</b>
<b>1</b>	<b>Introducción</b>	<b>23</b>
1.1	Objetivos . . . . .	23
1.2	Descripción . . . . .	23
1.3	Alcance . . . . .	24
<b>2</b>	<b>Metodología</b>	<b>25</b>
2.1	Marco teórico . . . . .	25
2.2	Tecnologías a utilizar . . . . .	25
2.2.1	Diseño de la arquitectura . . . . .	25
2.2.2	Diseño de componentes . . . . .	25
2.3	Análisis del sistema . . . . .	27
2.3.1	Hardware . . . . .	27
2.3.2	Software . . . . .	27
2.4	Diseño y desarrollo . . . . .	28
2.4.1	Hardware . . . . .	28
2.4.2	Software . . . . .	29
2.5	Pruebas del sistema . . . . .	29
2.5.1	Hardware . . . . .	29
2.5.2	Software . . . . .	29
<b>3</b>	<b>Conclusiones y trabajo futuro</b>	<b>31</b>
3.1	Conclusiones . . . . .	31
3.2	Trabajo futuro . . . . .	31

<b>4 Referencias/Bibliografía</b>	<b>33</b>
 <b>II Anexos técnicos</b>	 <b>35</b>
<b>A Manuales de usuario</b>	<b>37</b>
A.1 Clonación de tarjetas SD para tarjetas Zybo . . . . .	37
A.1.1 Clonación de Linux en tarjeta SD . . . . .	37
A.1.2 Inicio de Linux desde la tarjeta SD . . . . .	37
A.1.3 Creación de usuarios . . . . .	38
A.1.4 Habilitar SSH en placas Zybo . . . . .	39
A.2 Creación de una infraestructura de red de tarjetas Zybo . . . . .	41
A.2.1 Material necesario . . . . .	41
A.2.2 Pasos para el montaje de la infraestructura . . . . .	43
A.3 Test de interconexión de red Zybo . . . . .	48
A.3.1 Descripción . . . . .	48
A.4 Envío y recepción de ficheros . . . . .	50
A.4.1 Entre ordenador y tarjeta . . . . .	50
A.4.2 Entre tarjetas . . . . .	51
A.5 Comunicación y tratamiento de ficheros . . . . .	52
A.5.1 Introducción . . . . .	52
A.5.2 Directorios . . . . .	53
 <b>B Scripts</b>	 <b>55</b>
B.1 Inicio.sh . . . . .	56
B.1.1 Diagrama de flujo . . . . .	56
B.1.2 Código . . . . .	57
B.2 Lanzador.sh . . . . .	57
B.2.1 Diagrama de flujo . . . . .	58
B.2.2 Código . . . . .	58
B.3 Automatico.sh . . . . .	59
B.3.1 Diagrama de flujo . . . . .	59
B.3.2 Código . . . . .	59

<b>B.4</b>	<b>Recibiendo.sh</b>	60
B.4.1	Diagrama de flujo	60
B.4.2	Código	60
<b>B.5</b>	<b>Cristian.sh</b>	61
B.5.1	Diagrama de flujo	62
B.5.2	Código	63
<b>B.6</b>	<b>Enviando.sh</b>	64
B.6.1	Diagrama de flujo	65
B.6.2	Código	66
<b>B.7</b>	<b>Borrar.sh</b>	67
B.7.1	Diagrama de flujo	67
B.7.2	Código	67





# Lista de Figuras

2.1	Interfaces de red en la tarjeta Zybo . . . . .	28
A.1	Ejemplo de creación del usuario zybo0 . . . . .	38
A.2	Fichero sshd_config modificado . . . . .	39
A.3	Reiniciando el servicio SSH . . . . .	40
A.4	Placa Zybo Zynq 7010 . . . . .	41
A.5	Diagrama de Zybo Zynq 7010 procedente del manual de referencias . . . . .	42
A.6	Interfaces de red del ordenador central . . . . .	44
A.7	Configuración de PuTTY . . . . .	45
A.8	Interfaces de red de la tarjeta Zybo . . . . .	46
A.9	Líneas a modificar en el script Inicio.sh. . . . .	48
B.1	Diagrama de flujo de Inicio.sh . . . . .	56
B.2	Diagrama de flujo de Lanzador.sh. . . . .	58
B.3	Diagrama de flujo de Automatico.sh. . . . .	59
B.4	Diagrama de flujo de Recibiendo.sh. . . . .	60
B.5	Diagrama de flujo de Cristian.sh. . . . .	62
B.6	Diagrama de flujo de Diagramas/Enviando.sh. . . . .	65
B.7	Diagrama de flujo de Borrar.sh. . . . .	67



# Lista de Tablas

2.1 Direcciones IP de las tarjetas Zybo . . . . .	27
A.1 Direcciones IP de las tarjetas . . . . .	47



# **Parte I**

## **Contenido**



# Capítulo 1

## Introducción

### 1.1 Objetivos

El objetivo del trabajo es diseñar una red de nodos basada en tecnología ApSoC, de modo que cada uno de los nodos/elementos de la red reciban un fichero de datos, lo descifre, inserte información adicional y lo vuelva a cifrar antes de enviarlo a otro elemento de la red. El monitor generará el primer conjunto de datos que enviará a uno de los nodos, y cuando haya pasado por todos, recibirá el conjunto final. La red será privada y contará con un monitor basado en un ordenador personal.

### 1.2 Descripción

Cada uno de los nodos de la red será una tarjeta basada en la tecnología Zynq de Xilinx. Esta tecnología incluye un procesador ARM dual-core que se encargará de gestionar las comunicaciones en la red mediante protocolo TCP/IP. Para ello, se incluirá un sistema operativo basado en Linux.

El otro elemento constituyente de Zynq es lógica programable, en la que estará implementado el periférico cifrador/descifrador, ya diseñado y verificado en el trabajo de Fin de Grado de Cristian Ambrosio Costoya[1].

El diseño de la infraestructura de red implica:

1. La preparación e instalación de un arranque autónomo de cada tarjeta desde una memoria SD.
2. La interconexión física de las tarjetas y el monitor a través de un switch mediante una topología Ethernet, siendo el número de nodos ampliable de forma dinámica y automática.
3. Desarrollo de un conjunto de scripts para recibir, cifrar/descifrar, añadir información y enviar los ficheros de datos.
4. La creación y ejecución de un conjunto de pruebas que permitan confirmar el correcto funcionamiento de la red, en primera instancia, y el correcto funcionamiento del sistema de envío/recepción de datos, en segunda.

## 1.3 Alcance

El trabajo incluirá:

- Instalación física del ordenador personal que actuará como monitor.
- Creación de una imagen de arranque en tarjeta SD para las placas que formarán parte de la red. El arranque incluirá el bitstream necesario para configurar la lógica programable de Zynq con el IP AES core, así como el sistema operativo desarrollado por Gabriel Fernando Sánchez Reina [2] y los scripts.
- Instalación física y configuración de cada tarjeta en la red, y configuración del switch.
- Creación de los scripts necesarios para que cada nodo/tarjeta sea capaz de:
  - Recibir datos.
  - Descifre datos.
  - Añada datos.
  - Cifre datos.
  - Envíe el conjunto datos a otro nodo.
- Creación y ejecución de los tests que permitan comprobar el correcto funcionamiento de la infraestructura.
- Preparación del fichero de datos inicial en el monitor.
- Creación y ejecución de los tests que permitan comprobar el correcto funcionamiento de la transferencia, cifrado/descifrado y agregación de datos.



# Capítulo 2

## Metodología

### 2.1 Marco teórico

El punto de partida del proyecto consta de una serie de tarjetas Zybo [3] que incluyen la tecnología Zynq anteriormente citada, las cuales actuarán como nodos y que queremos conectar para realizar una comunicación entre ellas.

Dicha comunicación se establece para enviar un fichero entre ellas con el objetivo de recolectar una cierta información de cada una de ellas y enviarla al ordenador central.

### 2.2 Tecnologías a utilizar

#### 2.2.1 Diseño de la arquitectura

La infraestructura a diseñar es una red privada Ethernet TCP/IP y consta de los siguientes elementos:

- Tarjetas Zybo Zynq-7010 actuando como nodos de la red.
- Un ordenador con sistema operativo Linux (Debian 10 Buster)<sup>1</sup> y Windows 10. Será denominado “monitor” y será el encargado de enviar el fichero inicial de datos y recibir el fichero final.
- Un switch Tp-Link TL-SG1024D.

#### 2.2.2 Diseño de componentes

##### Ordenador monitor

El ordenador usado como monitor central es un Toshiba Satellite L750 que cuenta con los siguientes componentes:

- Procesador Intel Core I5-3240 quad-core.
- 6 Gb de memoria RAM.
- Disco duro SSD de 240 Gb.
- SO Debian 9 Stretch.

---

<sup>1</sup>Es posible usar cualquier distribución de Linux.

## Tarjetas Zybo Zynq 7010

Los componentes principales de este proyecto son las tarjetas Zybo Zynq 7010 que tienen las siguientes características:

- Procesador Cortex-A9 doble núcleo a 650 MHz.
- Memoria DDR3 con 8 canales de DMA<sup>2</sup>.
- Controladores de periféricos de gran ancho de banda: 1Gb Ethernet, USB 2.0.
- Controladores de periféricos de bajo ancho de banda: SPI, UART, CAN, I<sup>2</sup>C.
- Ranura MicroSD (compatible con el sistema de archivos Linux).
- Lógica reprogramable equivalente a Artix-7 FPGA:
  - 4.400 segmentos lógicos, cada uno con cuatro LUT de 6 entradas y 8 flip-flops.
  - 512 MB x32 DDR3 con ancho de banda de 1050 Mbps.
  - Dos cristales de administración de reloj, cada uno con un bucle de fase bloqueada (PLL) y un administrador de reloj de modo mixto (MMCM).
  - Procesadores digitales de señales de 80 componentes.
  - Velocidades de reloj interno que exceden los 450MHz.
  - Convertidor analógico a digital en chip (XADC).

## Switch

El switch utilizado es un Tp-Link TL-SG1024D con las siguientes especificaciones:

- Estándares y protocolos: IEEE 802.3i, IEEE 802.3u, IEEE 802.3ab , IEEE 802.3x.
- Interfaz: 24 puertos RJ45 a 10/100/1000 Mbps con negociación automática (MDI/MDIX automático).
- Medios de red: 10BASE-T: cable UTP categoría 3, 4, 5 (100 metros máximo) 100BASE-TX/1000BASE-T: cable UTP categoría 5, 5e o above cable (máximo 100 metros).
- Capacidad de conmutación: 48 Gbps.
- Tasa de reenvío de paquetes: 35.7 Mpps.
- Tabla de direcciones MAC: 8K.
- Certificaciones: FCC, CE, RoHS.

---

<sup>2</sup>Acceso Directo a Memoria: Permite a cierto tipo de componentes de una computadora acceder a la memoria del sistema para leer o escribir independientemente de la unidad central de procesamiento (CPU) principal.

## 2.3 Análisis del sistema

### 2.3.1 Hardware

- **Ordenador monitor:** Contará con un sistema operativo Linux. En este caso se ha usado Debian 9 Stretch y tendrá almacenado el fichero inicial de datos, que será modificado por el resto de nodos de la red, hasta que reciba el fichero final una vez que se complete la cadena. Es el punto de inicio y final de la cadena de nodos siendo tanto el emisor del fichero inicial como el receptor del fichero final (una vez que ha sido modificado por otros nodos de la red).

- **Nodos:** Cada tarjeta Zybo deberá incluir una instalación de un sistema operativo Linux para facilitar la gestión de los ficheros y las comunicaciones entre los nodos. El sistema operativo, el bitstream con el IP cifrador/descifrador y la aplicación forman una imagen desarrollada en el Trabajo de Fin de Grado de Gabriel Fernando Sánchez Reina [3]. Esta imagen, se clonará en las tarjetas de memoria SD incorporadas en cada tarjeta Zybo.

El proceso de clonación de las tarjetas SD lo podemos ver en el apéndice **Clonación de tarjetas SD para tarjetas Zybo**.

Para el mejor reconocimiento de las tarjetas Zybo en la red, cada una contará con un nombre y una IP fija.

Dispositivo	Dirección IP
monitor	192.168.1.10
zybo1	192.168.1.11
zybo2	192.168.1.12
zybo3	192.168.1.13

Tabla 2.1: Direcciones IP de las tarjetas Zybo

- **Switch:** Se ha usado un switch Tp-Link TL-SG1024D al que no se le ha aplicado ninguna configuración adicional.

### 2.3.2 Software

Para la transmisión del fichero de datos, usaremos SSH [4] debido a que nos ofrece la mayor seguridad a la hora de enviar ficheros a través de la red.

Se han diseñado una serie de scripts para automatizar el proceso de transmisión, cifrado/descifrado y adición de datos. Estos scripts<sup>3</sup> son los siguientes:

- **Inicio.sh:** Script encargado de probar las conexiones de todos los dispositivos de la red. Será lanzado manualmente para que el usuario pueda comprobar el estado de las conexiones.
- **Lanzador.sh:** Script encargado de lanzar el script `Automatico.sh` que contará con las correspondientes comprobaciones cíclicas. Éste script será lanzado mediante la herramienta `cron` al inicio del sistema operativo.
- **Automatico.sh:** Script encargado de lanzar periódicamente los siguientes scripts cada segundo para realizar las comprobaciones pertinentes. Debido a su comportamiento periódico, podemos garantizar la correcta comprobación de los directorios y ficheros implicados en el proceso.
- **Recibiendo.sh:** Script encargado de comprobar si se ha recibido algo y prepararlo para su tratamiento en cada nodo.
- **Cristian.sh:** Script encargado de descifrar los datos, añadir información y volver a cifrarlos en cada nodo. **Esto cambiará de nombre una vez que tenga lo de Gabri implementado.**

<sup>3</sup>Para ver detalladamente todos los scripts, ver el Apéndice B.

- **Enviando.sh:** Script encargado de enviar los datos al siguiente nodo. En caso de que no exista el siguiente nodo (o no esté disponible), se enviará de vuelta al ordenador central.
- **Borrar.sh:** Script encargado de borrar vaciar todos los directorios de trabajo. Deberá ser lanzado manualmente por el usuario en caso de querer vaciar los directorios de trabajo.

Cuando tenga todo completo, hay que añadir un diagrama de flujo para ver aquí como va la secuencia de scripts desde que llega un nuevo fichero a un nodo hasta que se envía al nuevo nodo.

## 2.4 Diseño y desarrollo

### 2.4.1 Hardware

Tanto las tarjetas Zybo como el ordenador monitor se conectarán al switch usando un cableado de red adecuado (podemos usar prácticamente cualquier tipo de cableado homologado, aunque, para este proyecto se han usado cables UTP categoría 5E).

Todos los dispositivos de la red tienen IP fija por lo que debemos configurar dichas IP en todos los ellos. Como estamos usando Linux, tendremos que configurar la IP en la interfaz correspondiente: Figura 2.1.

```

/dev/ttyUSB1 - PuTTY
zybo1@fhj:~$ su
Password:
root@fhj:/home/zybo1# ifconfig
eth0      Link encap:Ethernet  HWaddr 26:e6:3d:ca:06:b5
          inet addr:192.168.1.11  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::24e6:3dff:feca:6b5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:391 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:17046 (16.6 KiB)
          Interrupt:145 Base address:0xb000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:376 errors:0 dropped:0 overruns:0 frame:0
          TX packets:376 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:35576 (34.7 KiB)  TX bytes:35576 (34.7 KiB)

root@fhj:/home/zybo1#

```

Figura 2.1: Interfaces de red en la tarjeta Zybo

Para asociar una dirección IP al nombre de cada dispositivo, tendremos que rellenar el fichero `/etc/hosts` de todos los dispositivos con las direcciones IP y el nombre de los dispositivos conectados a la red. Este archivo será idéntico en todos los dispositivos para garantizar una consistente agenda de direcciones.

Este proyecto ha sido diseñado para que se puedan usar tantas tarjetas como se requiera o como permitan las capacidades físicas de la red ya que solo habrá que ajustar el fichero `/etc/hosts` para añadir más nodos en caso de que sea necesario.

Todo el proceso de asignación de IP's para la creación de la infraestructura de red, lo podemos ver en el anexo **Creación de una infraestructura de red de tarjetas Zybo**.

A la hora de hacer las pruebas del proyecto, hemos usado solo tres nodos cifradores/descifradores ya que es suficiente como para comprobar el correcto funcionamiento del mismo.

**Añadir un dibujo de la red con sus plaquitas, el switch, los cables y el ordenador de la forma más ordenada que pueda.**

## 2.4.2 Software

El proceso de comunicación se inicia en el ordenador monitor cuando éste envía un fichero a la primera tarjeta Zybo. Cuando ésta lo recibe, descifra el contenido, introduce los cambios necesarios, cifra el fichero y lo envía a la siguiente tarjeta.

Dicho proceso se lleva a cabo de la misma forma entre todas las tarjetas hasta que se llega a la última de éstas<sup>4</sup>, la cual envía el archivo nuevamente al ordenador monitor.

Toda esta comunicación se realiza gracias a la función de varios scripts programados en bash, por lo que tanto las tarjetas con Linux, como el ordenador monitor, son capaces de ejecutarlos.

Dichos scripts estarán automatizados mediante crontab<sup>5</sup>. Los podemos encontrar en el **Apéndice B** y se corresponden con la siguiente secuencia:

1. Recepción del fichero: Proceso llevado a cabo por el script **Recibiendo.sh**. Gracias al comando `stat` de Linux, podemos comprobar el estado del directorio de forma cíclica para poder detectar cualquier cambio (como la recepción de un fichero) en cuanto se produzca.
2. Descifrado, adición de datos y cifrado del fichero: Proceso llevado a cabo por el script **Cristian.sh**. **Esto cambiará de nombre una vez que tenga lo de Gabri implementado.**
3. Envío del fichero: Proceso llevado a cabo por el script **Enviando.sh**. Gracias al protocolo SSH y a la herramienta `sshpass`<sup>6</sup> de Linux, podremos enviar de forma automática el fichero modificado al siguiente nodo de la red.

En caso de que se quieran vaciar los directorios de trabajo, siempre se podrá lanzar de forma manual el script **Borrar.sh**.

## 2.5 Pruebas del sistema

Aquí describimos los scripts para hacer pruebas e incluir las pruebas que hice para ver su funcionamiento.

### 2.5.1 Hardware

**Prueba de la infraestructura: Lanzar el script Inicio.sh para que se vea que todas las tarjetas dan ping.**

### 2.5.2 Software

**Prueba de funcionamiento de la recolección de datos: Una prueba de todo funcionando.**

<sup>4</sup>Esto se puede ver en el archivo `/etc/hosts`, donde están almacenadas todas las direcciones IP de los dispositivos.

<sup>5</sup>Crontab es un administrador regular de procesos en segundo plano que ejecuta procesos o guiones a intervalos regulares.

<sup>6</sup>Esta herramienta es necesaria para poder funcionar automáticamente sin tener que aceptar manualmente la conexión con el servidor.



## Capítulo 3

# Conclusiones y trabajo futuro

### 3.1 Conclusiones

Después de las pruebas llevadas a cabo, podemos concluir el proyecto con un enfoque positivo. Gracias a la infraestructura de red creada, se ha conseguido con éxito una recopilación de datos colaborativa partiendo de un fichero inicial de datos desde el ordenador central, pasando por todos los nodos de la red y llegando, de nuevo, al ordenador central.

Se pueden destacar varios escenarios de trabajo: **Poner una imagen por cada escenario de trabajo.**

- Si todos los nodos están conectados correctamente a la red, se producirá un recorrido lineal que partirá desde el ordenador central, viajará por todos los nodos desde zybo1 hasta zyboX. Una vez que se recorra toda la cadena y, al no detectar el siguiente nodo, zybo(X+1), el fichero retornará al ordenador central.
- Cuando algún nodo intermedio de la red está desconectado, la cadena se romperá y, el nodo anterior, al no conseguir comunicación, enviará el fichero con la información recopilada al ordenador central, obviando el resto de nodos.
- Si el primer nodo de la red está desconectado, tendremos dos opciones:
  - Volver a revisar la conexión con el primer nodo hasta conseguir solucionar el problema.
  - Usar como primer nodo el segundo, enviando el fichero inicial directamente al segundo nodo de la red.
- **Creo que ahora sí que lo he entendido un poco mejor. Te refieres a que si la información “Hola 1”, se ha añadido en el primer nodo, que al lado ponga la ip del primer nodo y que si pone, “Hola 50”, al lado esté la ip del nodo 50, ¿correcto? Si es eso a lo que te refieres, sí que se podría hacer añadiendo la IP de cada nodo a lo que añade, pero claro, así sí que tendría sentido porque si se salta un nodo que está desconectado o lo que sea, sí que se podría saber de qué nodo procede cada información añadida al fichero. Ahora bien, ¿lo hago como contenido del proyecto (sería ajustar bastantes cosas) o lo pongo abajo como trabajo futuro?**

### 3.2 Trabajo futuro

Como trabajo futuro quedaría cambiar la cadena de conexiones y que, en vez de recorrer los nodos de forma secuencial, se hiciera de forma aleatoria, para que no se supiera qué nodo ha seguido a cual.

También se podría mejorar el proyecto incluyendo un módulo Wi-Fi, de modo que los nodos, no tengan que estar necesariamente conectados por cable. Esto nos daría la posibilidad de ubicar cada nodo donde quisiéramos (dentro de las capacidades físicas de la red Wi-Fi) y así poder usar dicha estructura en un entorno de IoT<sup>1</sup> para una casa o el edificio que necesitemos.

Este proyecto nos aporta una gran ventaja frente a la seguridad informática ya que, los ficheros transmitidos están doblemente cifrados, tanto por el propio protocolo SSH como por el IP cifrador/descifrador AES incorporado en los nodos Zybo. Esto hace que podamos recrearlo en cualquier red sin miedo a posibles filtraciones de datos.

---

<sup>1</sup>Internet of Things.



## Capítulo 4

# Referencias/Bibliografía

- [1] IP cifrador/descifrador como trabajo de fin de grado de Cristian Ambrosio Costoya.
- [2] Interfaz de conexión entre Linux y el IP cifrador/descifrador como trabajo de fin de grado de Gabriel Fernando Sánchez Reina.
- [3] Manual de referencia oficial de DIGILENT: [https://reference.digilentinc.com/\\_media/zybo:zybo\\_rm.pdf](https://reference.digilentinc.com/_media/zybo:zybo_rm.pdf).
- [4] Manual de SSH: <https://linux.die.net/man/1/ssh>.
- [5] Manual de la herramienta dd: <https://man7.org/linux/man-pages/man1/dd.1.html>
- [6] Manual de la herramienta sshpass: <https://linux.die.net/man/1/sshpas>.
- [7] Manual de scp: <https://linux.die.net/man/1/scp>.
- [8] Compresión y descompresión de ficheros <http://ecapy.com/comprimir-y-descomprimir-tgz-tar-gz-y-zip-por-linea-de-comandos-en-linux/index.html>.
- [9] Manual de stat: <https://linux.die.net/man/2/stat>.
- [10] Manual de crontab: <https://linux.die.net/man/1/crontab>.
- [11] Creación de diagramas de flujo gracias a la herramienta <https://app.diagrams.net/>.
- [12] Clonación de tarjetas SD en Linux: <https://www.altaruru.com/como-clonar-una-sd-raspberry-pi-orange-pi/>.
- [13] Protocolo SSH: [https://es.wikipedia.org/wiki/Secure\\_Shell](https://es.wikipedia.org/wiki/Secure_Shell)



## **Parte II**

### **Anexos técnicos**



## Apéndice A

# Manuales de usuario

### A.1 Clonación de tarjetas SD para tarjetas Zybo

#### A.1.1 Clonación de Linux en tarjeta SD

Como sistema operativo de las tarjetas Zybo, usaremos el que creó nuestro compañero Gabriel Fernando Sánchez Reina [2], el cual ya está preparado para usar el IP hardware de Cristian Ambrosio Costoya [1].

Para ello, seguiremos los siguientes pasos:

- Insertaremos la tarjeta SD con el sistema operativo en nuestro ordenador.
- Mediante la herramienta `dd` [5], guardaremos un fichero con extensión `.img` en nuestro ordenador, que será la copia de seguridad de la tarjeta SD.

```
sudo dd if=/dev/sdb of=/media/jesus/Gabri/Zybo.img
```

- A continuación, introducimos una tarjeta SD vacía de igual o mayor capacidad en nuestro ordenador.
- Mediante la herramienta `dd`, restauramos la copia de seguridad anterior en la nueva tarjeta SD.

```
sudo dd if=/media/jesus/Gabri/Zybo.img of=/dev/sdb
```

#### A.1.2 Inicio de Linux desde la tarjeta SD

Para iniciar la tarjeta con Linux debemos seguir los siguientes pasos:

- Insertar la tarjeta SD en la placa Zybo.
- Cambiar el jumper JP5 a la posición SD para que arranque desde dicha tarjeta SD.
- Conectamos el cable USB de la placa al ordenador y arrancamos la placa.
- Abrimos un terminal de comunicación serie, PuTTY en la consola del ordenador<sup>1</sup>. Ahora establecemos la conexión con la tarjeta Zybo y, podemos verificar la conexión y ver el arranque del sistema operativo.

---

<sup>1</sup>Puerto `ttyUSB1` y velocidad 115200.

### A.1.3 Creación de usuarios

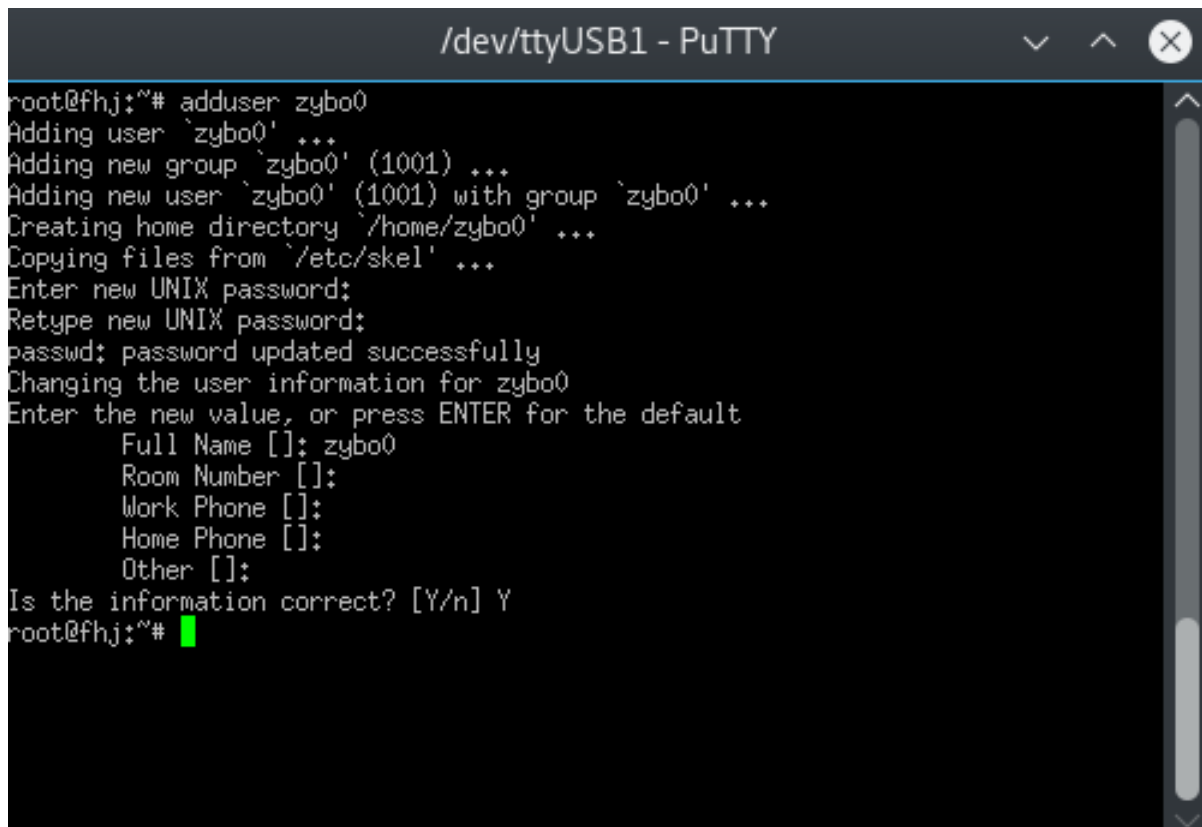
Al ser la primera vez que arrancamos el sistema operativo Linux, contamos únicamente con el usuario `root`, cuya contraseña es `root`. Por lo tanto, tenemos que crear otro usuario, que será con el que iniciemos sesión en las placas usando el comando:

```
adduser zyboX
```

Donde `X` es el identificador de la placa con la que estamos trabajando.

A continuación, tendremos que rellenar los siguientes campos:

- **Contraseña de usuario:** Introducimos la contraseña para el usuario creado. Si seguimos la nomenclatura que sigue el proyecto, será `zyboX`.
- **Repetir contraseña:** Repetiremos la contraseña para comprobar que no nos hemos equivocado.
- **Nombre:** Pondremos el nombre del usuario, `zyboX`, siguiendo la nomenclatura del proyecto.
- **Número de habitación, teléfono de trabajo y de casa, y “otro”:** Presionamos la tecla `ENTER` para dejarlo por defecto y continuar.



```
/dev/ttyUSB1 - PuTTY
root@fhj:~# adduser zybo0
Adding user `zybo0' ...
Adding new group `zybo0' (1001) ...
Adding new user `zybo0' (1001) with group `zybo0' ...
Creating home directory `/home/zybo0' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for zybo0
Enter the new value, or press ENTER for the default
    Full Name []: zybo0
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] Y
root@fhj:~#
```

Figura A.1: Ejemplo de creación del usuario `zybo0`

### A.1.4 Habilitar SSH en placas Zybo

Para establecer una conexión entre el ordenador central y las placas, tendremos que usar el protocolo SSH, que viene deshabilitado por defecto en Linux.

Para habilitarlo tendremos que acceder al archivo `/etc/ssh/sshd_config` como super-usuario. Para ello, utilizaremos el siguiente comando:

```
nano /etc/ssh/sshd_config
```

A continuación, nos dirigimos a la línea que tiene la siguiente sentencia:

```
#PasswordAuthentication yes
```

Borramos la almohadilla (#), guardamos y cerramos el fichero.

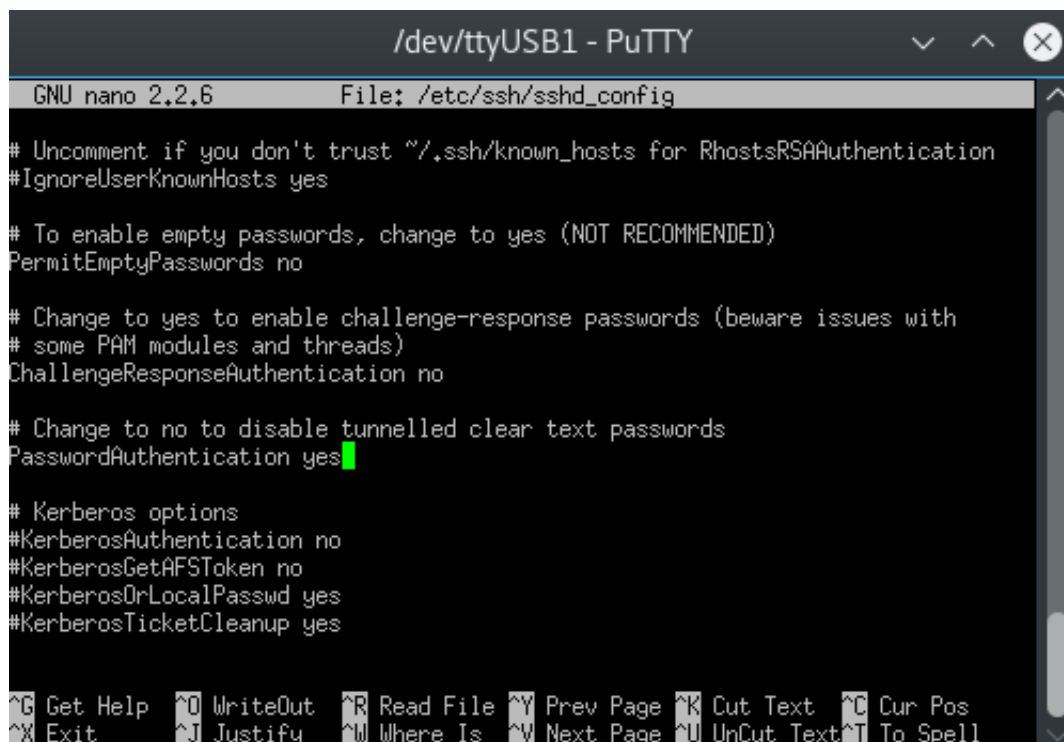
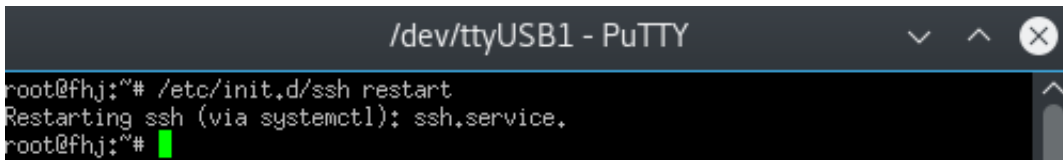


Figura A.2: Fichero `sshd_config` modificado

Para establecer los cambios realizados, debemos reiniciar el servicio SSH. Para ello utilizamos:

```
/etc/init.d/ssh restart
```

A screenshot of a PuTTY terminal window titled "/dev/ttyUSB1 - PuTTY". The terminal shows a root user at a machine named fhj. The user enters the command "/etc/init.d/ssh restart". The output of the command is "Restarting ssh (via systemctl): ssh.service.". The prompt returns to "root@fhj:~#" with a green cursor.

```
/dev/ttyUSB1 - PuTTY
root@fhj:~# /etc/init.d/ssh restart
Restarting ssh (via systemctl): ssh.service.
root@fhj:~#
```

Figura A.3: Reiniciando el servicio SSH

A partir de aquí ya podemos establecer conexiones SSH desde el ordenador central al resto de placas Zybo y enviar cualquier tipo de ficheros bien sea desde el ordenador central a las placas o bien, entre placas.



## A.2 Creación de una infraestructura de red de tarjetas Zybo

### A.2.1 Material necesario

Para la creación de la infraestructura de red física de placas Zybo contaremos con el siguiente material:

- Placas Zybo Zynq-7010.
- Un ordenador con sistema operativo Linux (Debian 9 Stretch)<sup>2</sup> y Windows 7.
- Un switch tp-link modelo TL-SG1024D.
- Software Vivado.

#### Placas Zybo Zynq-7000

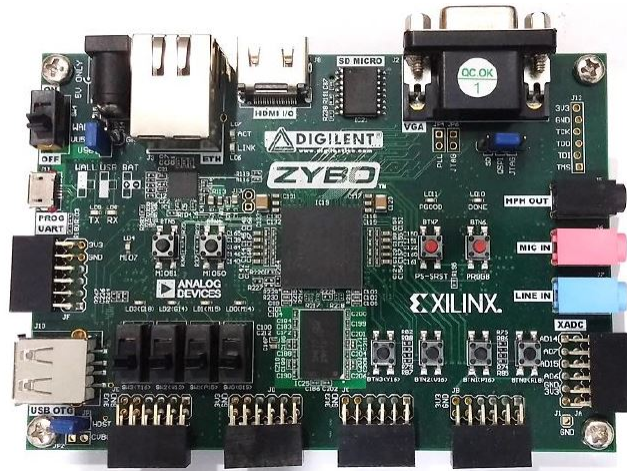


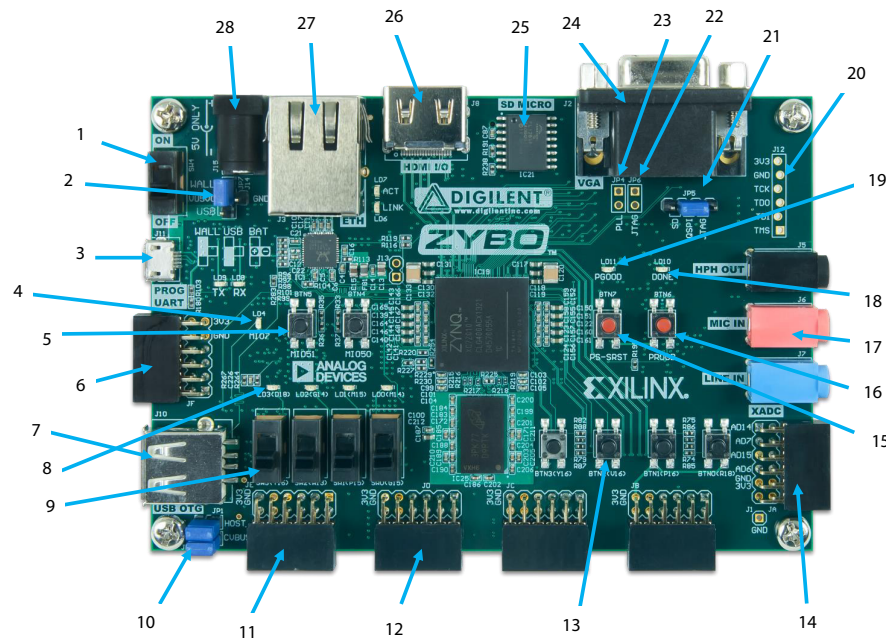
Figura A.4: Placa Zybo Zynq 7010

Para este proyecto necesitaremos poder programar la FPGA integrada en la placa desde la tarjeta SD de memoria. Para ello se va a preparar una imagen para que el procesador ARM integrado en la placa arranque desde la tarjeta SD y pueda programar la FPGA. El sistema operativo elegido es Linux.

Las placas Zybo Zynq 7010 tienen tres posibles modos de arranque que podemos seleccionar con el jumper JP5: QSPI, SD, JTAG. En este proyecto, el sistema operativo estará en la tarjeta SD, por lo tanto, tendremos que cambiar el jumper JP5 (situado arriba a la derecha) a la posición "SD"<sup>3</sup>.

<sup>2</sup>También es posible usar cualquier otra distribución de Linux.

<sup>3</sup>Dicho jumper está identificado con el número 21 en la imagen de la siguiente página.



Callout	Component Description	Callout	Component Description
1	Power Switch	15	Processor Reset Pushbutton
2	Power Select Jumper and battery header	16	Logic configuration reset Pushbutton
3	Shared UART/JTAG USB port	17	Audio Codec Connectors
4	MIO LED	18	Logic Configuration Done LED
5	MIO Pushbuttons (2)	19	Board Power Good LED
6	MIO Pmod	20	JTAG Port for optional external cable
7	USB OTG Connectors	21	Programming Mode Jumper
8	Logic LEDs (4)	22	Independent JTAG Mode Enable Jumper
9	Logic Slide switches (4)	23	PLL Bypass Jumper
10	USB OTG Host/Device Select Jumpers	24	VGA connector
11	Standard Pmod	25	microSD connector (Reverse side)
12	High-speed Pmods (3)	26	HDMI Sink/Source Connector
13	Logic Pushbuttons (4)	27	Ethernet RJ45 Connector
14	XADC Pmod	28	Power Jack

Figura A.5: Diagrama de Zybo Zynq 7010 procedente del manual de referencias

**Fuente:** Manual de referencia oficial de [DIGILENT®](https://www.digilentinc.com).

## Ordenador central

- **Sistemas operativos:** El ordenador usado en el proyecto tendrá dos sistemas operativos.
  - **Debian 9 Stretch:** Este sistema operativo tendrá un usuario llamado `zybo` y su contraseña será `zybomonitor`. La contraseña para los permisos de super-usuario también será `zybomonitor`. Este sistema operativo realizará la clonación del sistema operativo Linux a partir de la imagen generada en el Trabajo de Fin de Grado de Gabriel Fernando Sánchez Reina [2].
- **Red:** El ordenador central tendrá dos interfaces de red:
  - **Red externa:** Interfaz que nos proporcionará acceso a Internet en el ordenador central mediante la red de la UCA.
  - **Red interna:** Interfaz conectada al switch del proyecto y tendrá la IP 192.168.1.10 (estática).

## Switch

El switch usado en este proyecto es el modelo tp-link TL-SG1024D que cuenta con 24 puertos con tecnología Gigabit y conectores RJ-45. También cuenta con interfaz accesible para su configuración.

### A.2.2 Pasos para el montaje de la infraestructura

#### Asignar direcciones IP

Para asignarles las direcciones de IP a los dispositivos diferenciamos entre:

- **Ordenador:** Debemos identificar la interfaz de red con la que estamos trabajando. Para ello, abrimos un terminal y ejecutamos el siguiente comando como super-usuario<sup>4</sup>:

```
ifconfig
```

---

<sup>4</sup>Comando `su` y contraseña `zybomonitor`.

```

zybo@Monitor: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@Monitor:/home/zybo# ifconfig
enp4s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.142.14.78 netmask 255.255.0.0 broadcast 10.142.255.255
    inet6 fe80::221:70ff:fe3e:deca prefixlen 64 scopeid 0x20<link>
    ether 00:21:70:3e:de:ca txqueuelen 1000 (Ethernet)
    RX packets 1002726 bytes 292451083 (278.9 MiB)
    RX errors 0 dropped 247 overruns 0 frame 0
    TX packets 15307 bytes 1944983 (1.8 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp5s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether 00:13:f7:71:b0:a3 txqueuelen 1000 (Ethernet)
    RX packets 542 bytes 33166 (32.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 281 bytes 34689 (33.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Local Loopback)
    RX packets 12821 bytes 1070302 (1.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 12821 bytes 1070302 (1.0 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@Monitor:/home/zybo#

```

Figura A.6: Interfaces de red del ordenador central

En la figura A.6, podemos ver las siguientes interfaces:

- **enp4s0**: Interfaz de red externa.
- **enp5s0**: Interfaz de red interna.

Una vez identificada la interfaz, debemos acceder al fichero

/etc/network/interfaces.d/INTERFAZ<sup>5</sup> como super-usuario con el siguiente comando:

```
gedit /etc/network/interfaces.d/enp5s0
```

Y lo modificamos de la siguiente forma:

```

1 allow-hotplug enp5s0
2     iface enp5s0 inet static
3         address 192.168.1.10
4         netmask 255.255.255.0
5         gateway 192.168.1.1

```

<sup>5</sup>Siendo INTERFAZ, la interfaz que estamos usando. En este ejemplo, enp5s0.

- **Tarjetas Zybo:** Conectamos la tarjeta mediante USB al ordenador y abrimos un terminal serie en PuTTY<sup>6</sup> e iniciamos sesión en Linux<sup>7</sup>.

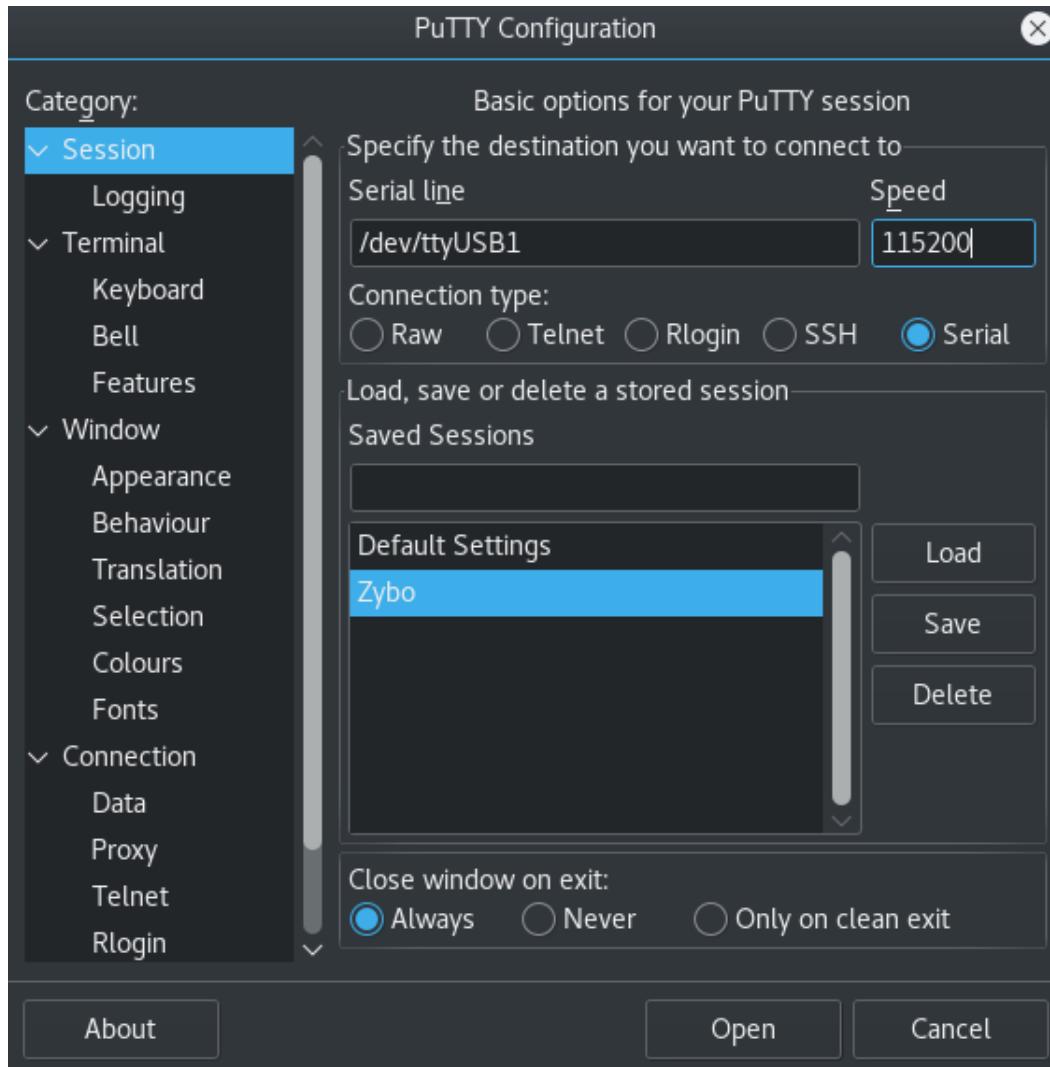


Figura A.7: Configuración de PuTTY

Debemos identificar la interfaz de red con la que estamos trabajando. Para ello, ejecutamos el siguiente comando como super-usuario<sup>8</sup>:

```
ifconfig
```

<sup>6</sup>Puerto ttyUSB1 y velocidad 115200.

<sup>7</sup>Usuario: zyboX; contraseña zyboX. Siendo X el identificador de la tarjeta.

<sup>8</sup>Comando su y contraseña root.

```

/dev/ttyUSB1 - PuTTY
zybo1@fhj:~$ su
Password:
root@fhj:/home/zybo1# ifconfig
eth0      Link encap:Ethernet  HWaddr 26:e6:3d:ca:06:b5
          inet addr:192.168.1.11  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::24e6:3dff:feca:6b5/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:391 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:17046 (16.6 KiB)
          Interrupt:145 Base address:0xb000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128  Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:376 errors:0 dropped:0 overruns:0 frame:0
          TX packets:376 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:35576 (34.7 KiB)  TX bytes:35576 (34.7 KiB)

root@fhj:/home/zybo1#

```

Figura A.8: Interfaces de red de la tarjeta Zybo

Una vez identificada la interfaz, debemos acceder al fichero

/etc/network/interfaces.d/INTERFAZ<sup>9</sup> como super-usuario con el siguiente comando:

```
nano /etc/network/interfaces.d/eth0
```

Y lo modificamos de la siguiente forma:

```

1  allow-hotplug eth0
2      iface eth0 inet static
3      address 192.168.1.11
4      netmask 255.255.255.0
5      gateway 192.168.1.1

```

<sup>9</sup>Siendo INTERFAZ, la interfaz que estamos usando. En este ejemplo eth0.

Para establecer la dirección IP del resto de dispositivos, repetiremos el proceso y estableceremos las direcciones IP siguiendo la siguiente tabla:

Dispositivo	Dirección IP
Monitor	192.168.1.10
Zybo1	192.168.1.11
Zybo2	192.168.1.12
Zybo3	192.168.1.13
Zybo4	192.168.1.14

Tabla A.1: Direcciones IP de las tarjetas

Las tarjetas estarán identificadas como ZyboX (siendo “X” el identificador de la tarjeta con la que estamos trabajando) y el ordenador se identificará como “Monitor”.

Se puede observar que el número identificador de la tarjeta coincide con el último número de la dirección IP –10, de tal forma que si queremos añadir la tarjeta Zybo10, su dirección IP será 192.168.1.20.

### Conexión al switch de las placas Zybo

Una vez tengamos los dispositivos identificados tenemos que conectarlos al switch<sup>10</sup>. No es necesario una configuración previa del switch ni entrar en su interfaz web, ya que no vamos a requerir acciones avanzadas.

Para probar la conectividad entre todos los dispositivos tendremos que ejecutar el test de interconexión de red. Dicho test se encuentra descrito en el anexo **Test de interconexión de red Zybo**.

<sup>10</sup>Podemos conectar los dispositivos al puerto del switch que queramos debido a que se encargará de ir rellenando su tabla CAM con las direcciones de los dispositivos que tiene conectados.

## A.3 Test de interconexión de red Zybo

### A.3.1 Descripción

Una vez se han conectado todos los dispositivos al switch como se indicó en el tutorial de creación de una infraestructura de red de placas Zybo, y, estando encendidos todos los dispositivos, necesitamos comprobar que existe conexión entre todos ellos.

Para ello, se ha desarrollado un script en bash, llamado `Inicio.sh`, que comprueba el estado de conexión de los dispositivos. En este script se pueden definir las tarjetas Zybo que se vayan a usar, con su alias y su dirección IP de red (previamente establecida). Para ello, tendremos que abrir el fichero `Inicio.sh` con nuestro editor favorito y cambiar las siguientes líneas:

```
3 direcciones=(192.168.1.11 192.168.1.12 192.168.1.13 192.168.1.14)
4 tarjetas=(zybo1 zybo2 zybo3 zybo4)
```

Figura A.9: Líneas a modificar en el script `Inicio.sh`.

- **Línea 3:** Introducir la dirección IP de la tarjeta que queramos añadir. O borrar la que queramos quitar.
- **Línea 4:** Introducir el alias de la tarjeta que queramos añadir. O borrar la que queramos quitar.

El script será ejecutado desde el ordenador central abriendo una terminal en el directorio donde se ubique el mismo. Este script lo podemos encontrar en el apéndice [Inicio.sh](#).

Cabe destacar que tiene dos modos de funcionamiento:

#### Normal

En esta opción, la salida del script será meramente informativa, devolviendo si la tarjeta está o no conectada a la red.

- Se ejecuta mediante el comando:

```
./Inicio.sh
```

- El script lanza cuatro paquetes de ping a cada una de las tarjetas, una por una, guardando el resultado del mismo comando en una variable y sin mostrarla por pantalla.
- Mediante scraping<sup>11</sup> el script lee la salida del último de los paquetes de ping.
- **Éxito:** Si en dicho texto no se encuentra la secuencia “100% packet loss” indicará que la tarjeta está conectada.
- **Fracaso:** Si en dicho texto se encuentra la secuencia “100% packet loss” indicará que la tarjeta a la cual se está realizando el ping, no está conectada.

<sup>11</sup>Técnica usada mediante programas software para extraer información de un texto.



### Verboso

En esta opción, la salida del script es más detallada ya que muestra más información al respecto de la conexión.

- Se ejecuta mediante el comando:

```
./Inicio.sh -v
```

- El script lanza cuatro paquetes de ping a cada una de las tarjetas, una por una.
- Mediante la opción `-v` le indicamos que nos muestre por pantalla la salida de los comandos de ping, para ver el estado de la conexión más detalladamente.

## A.4 Envío y recepción de ficheros

Para el envío y recepción de archivos entre los distintos dispositivos, usaremos la utilidad `sshpass` [7] que está diseñada para ejecutar `ssh` de modo no-interactivo.

Para instalar `sshpass` lo haremos de la siguiente forma tanto en el ordenador central como en las tarjetas zybo:

1. Entramos en modo super-usuario con el comando: `su`<sup>12</sup>.
2. Introducimos el siguiente comando para la instalación de `sshpass`:

```
apt-get install sshpass
```

### A.4.1 Entre ordenador y tarjeta

Para el envío de archivos, independientemente de su extensión, mediante `ssh`<sup>13</sup> desde el ordenador a las tarjetas Zybo debemos usar el siguiente comando en un terminal del ordenador ubicado en el directorio donde está el archivo que queremos enviar:

```
sshpass -p zyboX scp -o StrictHostKeyChecking=no archivoLocal  
zyboX@zyboX:/home/zyboX/ficheros/recibir
```

Siendo:

- **zyboX:** La tarjeta Zybo a la que queremos enviar el archivo<sup>14</sup>. Por ejemplo: `zybo1`.
- **archivoLocal:** Nombre del archivo local que queremos enviar.
- **Directorio /ficheros/recibir:** Directorio donde se recibirán los archivos en el proyecto<sup>15</sup>.

---

<sup>12</sup>La contraseña será `zybomonitor` (para el ordenador central), o `root` (para las tarjetas).

<sup>13</sup>La opción “`-o StrictHostKeyChecking=no`” que encontraremos en los siguientes comandos se usa para evadir la confirmación de conexión y que la acepte inmediatamente.

<sup>14</sup>Gracias a la existencia del fichero `/etc/hosts` tenemos asociada cada tarjeta con su dirección de red. Por lo tanto, solo tenemos que poner el alias de la tarjeta para referirnos a su dirección IP.

<sup>15</sup>Si queremos enviar un fichero a otra ubicación, solo debemos cambiar la ruta donde queremos enviarlo.

### A.4.2 Entre tarjetas

Para el envío de archivos entre las tarjetas Zybo tenemos dos formas, manual y automática:

#### Manual

Debemos conectarnos a las placas por SSH, desde el ordenador central, usando el siguiente comando:

```
sshpass -p zyboX ssh -o StrictHostKeyChecking=no zyboX@zyboX
```

Donde X es el identificador de la placa a la que nos queremos conectar.

Luego, nos situamos en el directorio donde se encuentra el archivo de la primera placa que queramos enviar a la segunda, y escribimos el siguiente comando:

```
sshpass -p zyboX scp -o StrictHostKeyChecking=no archivoLocal  
zyboX@zyboX:/home/zyboX/ficheros/recibir
```

Siendo:

- **zyboX:** La tarjeta Zybo a la que queremos enviar el archivo. Por ejemplo: zybo1.
- **archivoLocal:** Nombre del archivo local que queremos enviar.
- **Directorio /ficheros/recibir:** Directorio donde se recibirán los archivos en el proyecto<sup>16</sup>.

#### Automática

Usaremos el script `Enviando.sh` situado en el directorio `~/ficheros` de las placas Zybo. Este script lo podemos encontrar en el apéndice `Enviando.sh`

Este script usará el mismo comando que el scp [7] de forma manual, pero estará parametrizado para que lo envíe al siguiente dispositivo.

---

<sup>16</sup>Si queremos cambiar el directorio, solo tenemos que cambiarlo al igual que en el caso ordenador-placa.

## A.5 Comunicación y tratamiento de ficheros

En este documento se desarrolla un tutorial de envío y recepción de ficheros mediante SSH entre los dispositivos de la red de manera secuencial y automática.

Para conseguir dicha finalidad, las tarjetas iniciarán automáticamente un proceso que comprobará si les envían un fichero nuevo y, en caso de que así sea, realizarán alguna modificación en él y lo enviarán a la siguiente tarjeta, o, en caso de que dicha tarjeta sea la última, al ordenador central.

Particularmente, la tarea a realizar aquí consta de la recepción de un fichero cifrado, su descifrado, modificación, cifrado y, por último, su envío hacia el siguiente dispositivo.

Se describirá una estructura de directorios que serán recorridos por el fichero inicial en sus distintos estados. Basándonos en dichos estados, nombraremos los directorios tal como se describen a continuación:

- `/ficheros/recibir`: Directorio donde se recibirán los ficheros.
- `/ficheros/desencriptar`: Directorio donde se dejan los ficheros una vez desencriptados.
- `/ficheros/trabajar`: Directorio donde se dejan los ficheros para incluir los datos y volverlo a cifrar.
- `/ficheros/enviar`: Directorio donde se dejan los ficheros que están listos para ser enviados al siguiente dispositivo.

### A.5.1 Introducción

Para realizar una comunicación automática y secuencial entre los distintos dispositivos, cada tarjeta Zybo Zynq 7010 contará con la siguiente estructura de directorios:

```
~/ficheros/backups/
|
|      |ViejoEnviar.txt
|      |ViejoRecibir.txt
|      |ViejoTrabajar.txt
|desencriptar/..
|enviar/..
|recibir/..
|trabajar/..
|Automatico.sh
|Borrar.sh
|Cristian.sh
|Enviando.sh
|Recibiendo.sh
```

Este árbol de directorios cambiará una vez que tenga los nuevos nombres del script de Gabri y lo que me haga falta.

Este árbol de directorios estará en el archivo `ficheros.tar.gz` que se encontrará en el ordenador central y será distribuido a cada tarjeta mediante `ssh`<sup>17</sup> con el siguiente comando:

```
sshpass -p zyboX scp -o StrictHostKeyChecking=no ficheros.tar.gz zyboX@zyboX:
```

<sup>17</sup>Recordemos que, tanto para `ssh` como para `scp`, el elemento `zyboX` es el identificador de la tarjeta Zybo con la que estamos trabajando.

Luego, entramos en la tarjeta mediante ssh con el siguiente comando:

```
sshpass -p zyboX ssh -o StrictHostKeyChecking=no zyboX@zuboX
```

Para ver si tenemos el archivo `ficheros.tar.gz` usamos el comando:

```
ls
```

A continuación, aplicamos el siguiente comando para descomprimir [8] el archivo `ficheros.tar.gz`:

```
tar -xzvf ficheros.tar.gz
```

Se nos creará el árbol de directorios anteriormente citado en el directorio `/home` de la tarjeta Zybo a la que hayamos accedido.

Para que la automatización del proceso se lleve a cabo correctamente, también tendremos que modificar el fichero `/etc/hosts` de la tarjeta. Para ello, enviamos el fichero con el siguiente comando:

```
sshpass -p zyboX scp -o StrictHostKeyChecking=no hosts zyboX@zyboX:
```

Y, luego, lo copiamos como super-usuario<sup>18</sup> en el directorio `/etc` con el siguiente comando:

```
cp hosts /etc
```

Una vez hecho esto, el proceso de automatización estaría listo para ser lanzado.

## A.5.2 Directorios

En este apartado, describiremos los distintos directorios que podemos encontrar en las tarjetas Zybo una vez que se ha descomprimido el archivo `ficheros.tar.gz`.

Dicha descripción se hará siguiendo el orden que recorrerá el archivo recibido desde el ordenador central, salvo el directorio `/backups` que se describirá primero ya que no interviene de forma directa en el camino a recorrer por el archivo recibido.

### Directorio `/backups`

En este directorio se guardarán los últimos estados de los directorios nombrados a continuación, generados por el comando `stat` [9].

- **ViejoDesencriptar.txt:** Este fichero contendrá el estado del directorio `/desencriptar` una vez que el script `Desencriptando.sh` lo compruebe. Si no se producen cambios en dicho directorio, este fichero no se modificará.
- **ViejoEnviar.txt:** Este fichero contendrá el estado del directorio `/enviar` una vez que el script `Enviando.sh` lo compruebe. Si no se producen cambios en dicho directorio, este fichero no se modificará.
- **ViejoRecibir.txt:** Este fichero contendrá el estado del directorio `/recibir` una vez que el script `Recibiendo.sh` lo compruebe. Si no se producen cambios en dicho directorio, este fichero no se modificará.
- **ViejoTrabajar.txt:** Este fichero contendrá el estado del directorio `/trabajar` una vez que el script `Trabajando.sh` lo compruebe. Si no se producen cambios en dicho directorio, este fichero no se modificará.

---

<sup>18</sup>Comando: `su`, y contraseña `root`.

**Directorio /recibir**

Este directorio contendrá los ficheros enviados por otros dispositivos mediante ssh<sup>19</sup>.

**Directorio /desencriptar**

Este directorio contendrá los ficheros enviados desde el directorio /recibir por el script `Recibiendo.sh` que comprobará el estado del directorio anterior.

**Directorio /trabajar**

Este directorio contendrá los ficheros enviados por el script `Cristian.sh` que comprobará el estado del directorio anterior.

Será donde se realicen las acciones específicas que se contemplan en este tutorial.

**Directorio /enviar**

Este directorio contendrá los ficheros enviados por el script `Cristian.sh` que comprobará el estado del directorio anterior.

Una vez aquí, dichos ficheros estarán listos para pasar al siguiente dispositivo.

---

<sup>19</sup>Para ver como enviar ficheros desde un dispositivo a otro hasta este directorio, ver el documento “Envío y recepción de ficheros con sshpass”.

## Apéndice B

# Scripts

En este apéndice, describiremos los distintos scripts que podemos encontrar en las tarjetas Zybo con su explicación y código correspondiente.

Dicha descripción se hará siguiendo el orden que recorrerá el archivo recibido desde el dispositivo anterior hasta ser enviado al siguiente dispositivo.

Para comprobar el estado de todos los directorios, usaremos el comando `stat` para comprobar el estado de los directorios.

En el trabajo aquí mencionado se emula el descriptado de un fichero, adición de información, cifrado, y envío del mismo a otro dispositivo<sup>1</sup>. **Esto habría que cambiarlo puesto que ahora sí que tenemos el trabajo de Cristian.**

---

<sup>1</sup>Para cambiar dicho comportamiento, solo tendremos que modificar los scripts que se encargan de automatizar el proceso.

## B.1 Inicio.sh

El script se encarga de probar las conexiones de todos los dispositivos de la red para ver que están conectados al switch.

### B.1.1 Diagrama de flujo

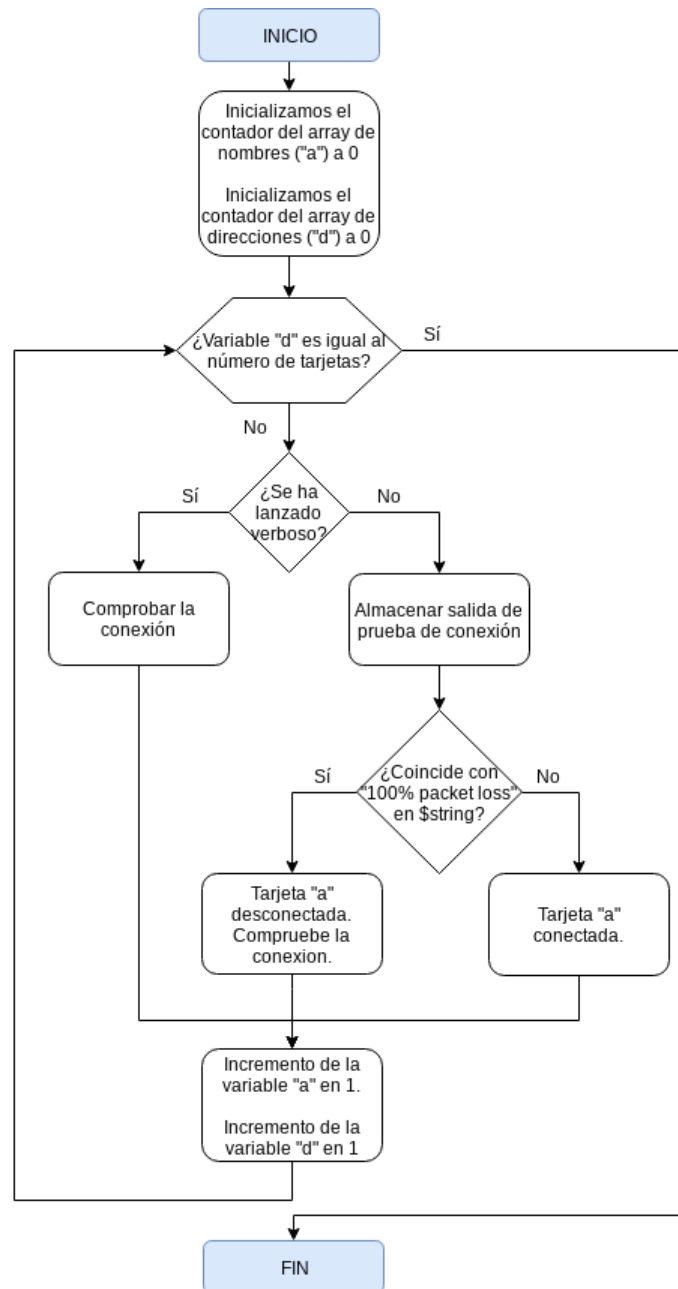


Figura B.1: Diagrama de flujo de `Inicio.sh`



## B.1.2 Código

```

1  #!/bin/bash
2
3  #Inicio.sh
4
5  direcciones=(192.168.1.2 192.168.1.3 192.168.1.4 192.168.1.5)
6  tarjetas=(zybo1 zybo2 zybo3 zybo4)
7  a=0
8  for d in "${direcciones[@]}"
9  do
10   if [[ $1 = "-v" ]]; then
11     echo Comprobando la conexión con ${tarjetas[a]}.
12     ping -c 4 $d
13   else
14     echo Comprobando la conexión con ${tarjetas[a]}.
15     string=$(ping -c 4 $d)
16     if [[ $string == *100%\ packet\ loss* ]]; then
17       echo Tarjeta ${tarjetas[a]} desconectada. Compruebe la conexión.$'\n'
18     else
19       echo Tarjeta ${tarjetas[a]} conectada.$'\n'
20     fi
21   fi
22   let a=a+1
23 done

```

Código de Inicio.sh usando cuatro tarjetas como ejemplo.

## B.2 Lanzador.sh

Este script se encarga de lanzar el script Automatico.sh mediante la herramienta crontab [10] al inicio del sistema operativo Linux.

Para usarlo, debemos usar el siguiente comando:

```
crontab -e
```

Y, luego, añadir la regla que queramos que se ejecute al final del fichero. En nuestro caso es la siguiente:

```
@reboot (cd ~/ficheros; ./Lanzador.sh)
```

Esto hará que la herramienta cron inicie este script al iniciar el sistema operativo Linux de las tarjetas Zybo.

### B.2.1 Diagrama de flujo

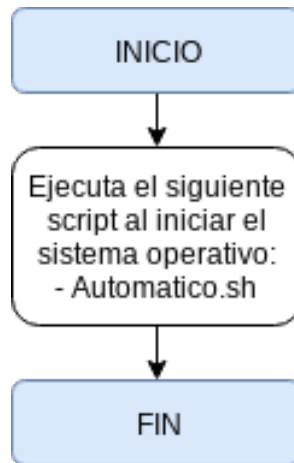


Figura B.2: Diagrama de flujo de `Lanzador.sh`.

### B.2.2 Código

```
1 #!/bin/bash
2
3 #Lanzador.sh
4
5 ./Automatico.sh
```

Código de `Lanzador.sh`.

## B.3 Automatico.sh

Este script es el encargado de lanzar el resto de scripts periódicamente para que vayan comprobando los directorios correspondientes y se produzca la comunicación de forma automática.

### B.3.1 Diagrama de flujo

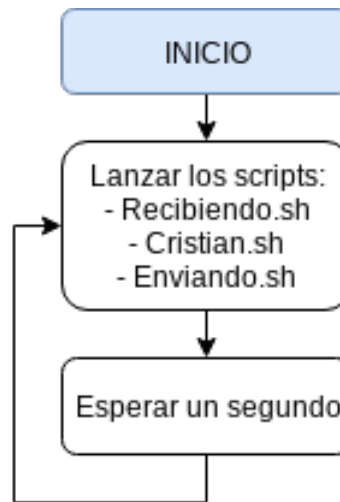


Figura B.3: Diagrama de flujo de Automatico.sh.

### B.3.2 Código

```
1 #!/bin/bash
2
3 #Automatico.sh
4
5 while :
6 do
7     ./Recibiendo.sh
8     ./Cristian.sh
9     ./Enviando.sh
10    sleep 1
11 done
```

Código de Anexos/Anexo3/Automatico.sh.

## B.4 Recibiendo.sh

Este script es el encargado de comprobar el estado del directorio ~/ficheros/recibir y, si llega un archivo nuevo, enviarlo al directorio ~/ficheros/desencriptar.

### B.4.1 Diagrama de flujo

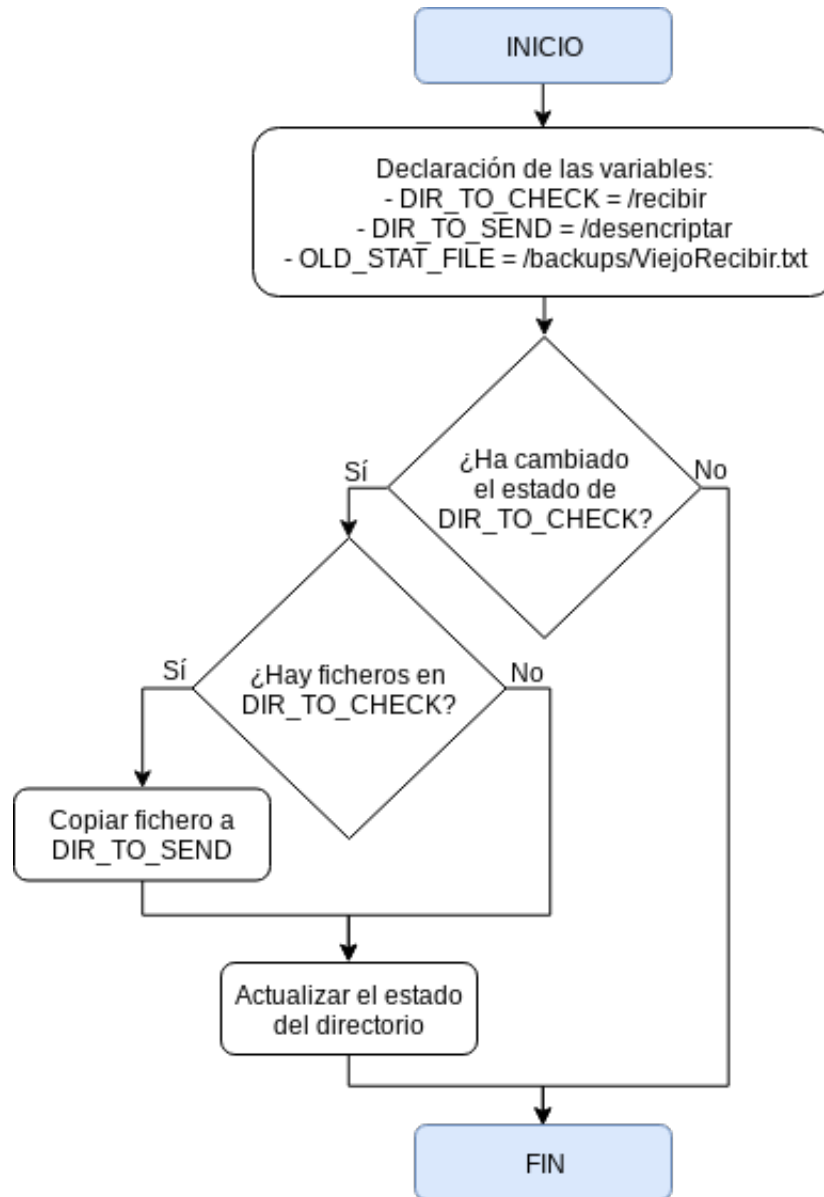


Figura B.4: Diagrama de flujo de `Recibiendo.sh`.

### B.4.2 Código

```

1  #!/bin/bash
2

```

```

3 #Recibiendo.sh
4
5 DIR_TO_CHECK=$PWD/recibir #Directorio que queremos inspeccionar
6
7 DIR_TO_SEND=$PWD/desencriptar #Directorio donde enviar
8
9 OLD_STAT_FILE=$PWD/backups/ViejoRecibir.txt #Estado del directorio
10
11 if [ -e $OLD_STAT_FILE ]
12 then
13     OLD_STAT=`cat $OLD_STAT_FILE`
14 else
15     OLD_STAT="nothing"
16 fi
17
18 NEW_STAT=`stat -t $DIR_TO_CHECK`
19
20 if [ "$OLD_STAT" != "$NEW_STAT" ]
21 then
22
23     cd $DIR_TO_CHECK
24
25     n=$(ls | wc -l)
26     cero=0
27
28     if [[ n -ne cero ]]
29     then
30         fichero=$(ls -t | head -1) #Obtenemos el fichero más reciente
31         cp $fichero $DIR_TO_SEND
32     fi
33     echo $NEW_STAT > $OLD_STAT_FILE #Actualiza el estado del directorio
34 fi

```

Código de Recibiendo.sh.

## B.5 Cristian.sh

A este script le pondré el nombre `Encriptando.sh` seguramente, porque básicamente se encargará de eso (tanto encriptar como desencriptar), o, a unas malas, le pongo `Trabajando.sh`. También tengo la opción de hacer dos, uno que encripte y otro que desencripte, depende de como vea que funciona una vez que tenga las tarjetas. Así que, hasta que no tenga las tarjetas y me ponga a hacer las pruebas de verdad con todo (tanto lo de Gabri como lo de Cristian, esta parte queda un poco en el aire para perfeccionarla luego.).

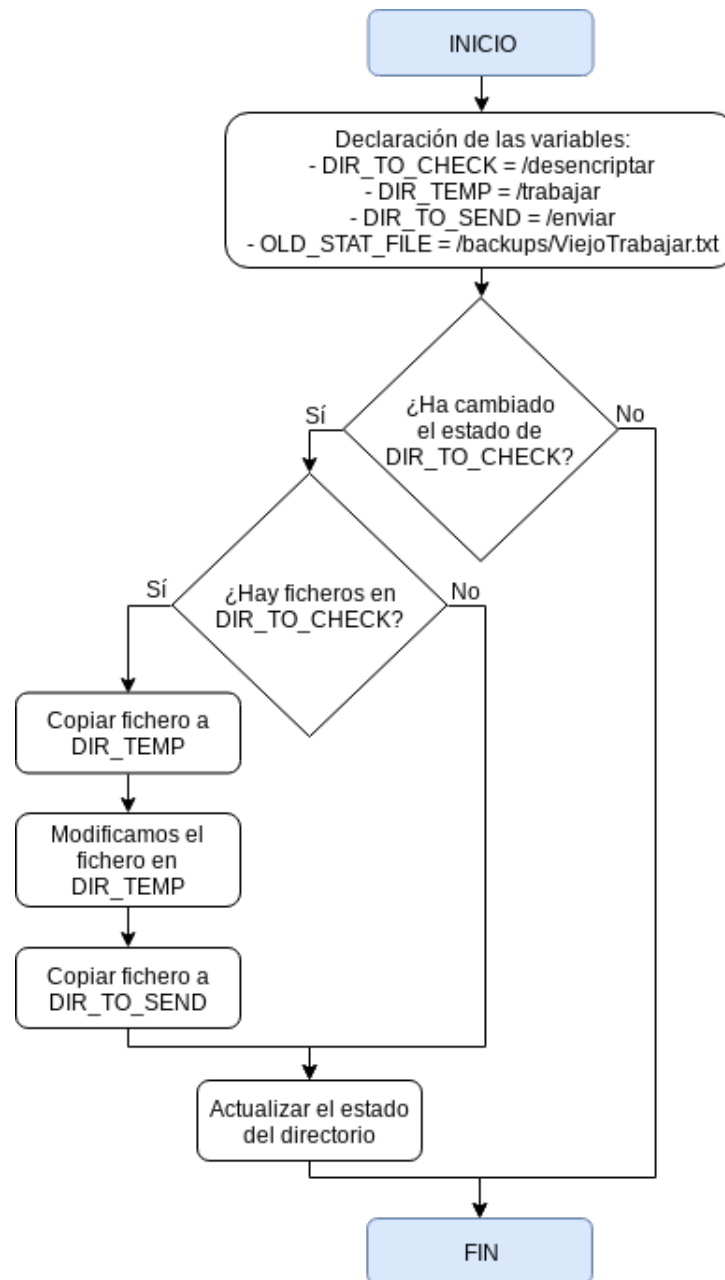
Este script es el encargado de emular el trabajo de nuestro compañero Cristian y realiza las siguientes tareas:

- Gracias al crontab establecido, se encarga de comprobar periódicamente el estado del directorio `~/ficheros/desencriptar`.
- Mueve el archivo allí situado al directorio `~/ficheros/trabajar` (simulando el desencriptado del mismo).
- Una vez allí, añade un texto como el siguiente:

Archivo tratado en zyboX

Siendo zyboX el identificador de la tarjeta con la que estamos trabajando.

- Por último, envía el fichero al directorio `~/ficheros/enviar`.

**B.5.1 Diagrama de flujo**Figura B.5: Diagrama de flujo de `Cristian.sh`.

## B.5.2 Código

```

1  #!/bin/bash
2
3  #Cristian.sh
4
5  DIR_TO_CHECK=$PWD/desencriptar #Directorio que queremos inspeccionar
6
7  DIR_TEMP=$PWD/trabajar #Directorio temporal para trabajar
8
9  DIR_TO_SEND=$PWD/enviar #Directorio donde enviar
10
11 OLD_STAT_FILE=$PWD/backups/ViejoTrabajar.txt #Estado del directorio
12
13 if [ -e $OLD_STAT_FILE ]
14 then
15     OLD_STAT=`cat $OLD_STAT_FILE`
16 else
17     OLD_STAT="nothing"
18 fi
19
20 NEW_STAT=`stat -t $DIR_TO_CHECK`
21
22 if [ "$OLD_STAT" != "$NEW_STAT" ]
23 then
24
25     cd $DIR_TO_CHECK
26
27     n=$(ls | wc -l)
28     cero=0
29
30     if [[ n -ne cero ]]
31     then
32         fichero=$(ls -t | head -1) #Obtenemos el fichero más reciente
33         cp $fichero $DIR_TEMP
34         cd $DIR_TEMP
35         tarjeta=$(cat /etc/passwd | cut -d : -f1 | grep zybo)
36         echo 'Archivo_tratado_en_'$tarjeta >> $fichero
37         cp $fichero $DIR_TO_SEND
38     fi
39     echo $NEW_STAT > $OLD_STAT_FILE #Actualiza el estado del directorio
40 fi

```

Código de Cristian.sh.

## B.6 Enviando.sh

Este script es el encargado de comprobar periódicamente el estado del directorio `~/ficheros/enviar` y, cuando detecta un cambio, envía el archivo a la siguiente tarjeta, o, si ésta se encuentra desconectada, al ordenador central.

A la hora de comprobar si la siguiente tarjeta está conectada o no, se hace enviando un comando `ping` a la siguiente tarjeta, por lo que se nos presentarán dos posibles casos:

- **Éxito:** La tarjeta está conectada y será allí donde se envíe el fichero.
- **Fracaso:** La tarjeta no está conectada y el fichero será enviado al ordenador central.

Para que podamos usar el comando `ping` desde este script, debemos darle permisos de ejecución en modo usuario de la siguiente forma:

1. Entramos como super-usuario con el comando `su` y contraseña `zyboX` (siendo `X` el identificador de la tarjeta con la que estamos trabajando).
2. A continuación, introducimos el siguiente comando:

```
chmod u+s /bin/ping
```

Y con eso, quedaría activado el comando `ping` para poder usarlo desde este script.



### B.6.1 Diagrama de flujo

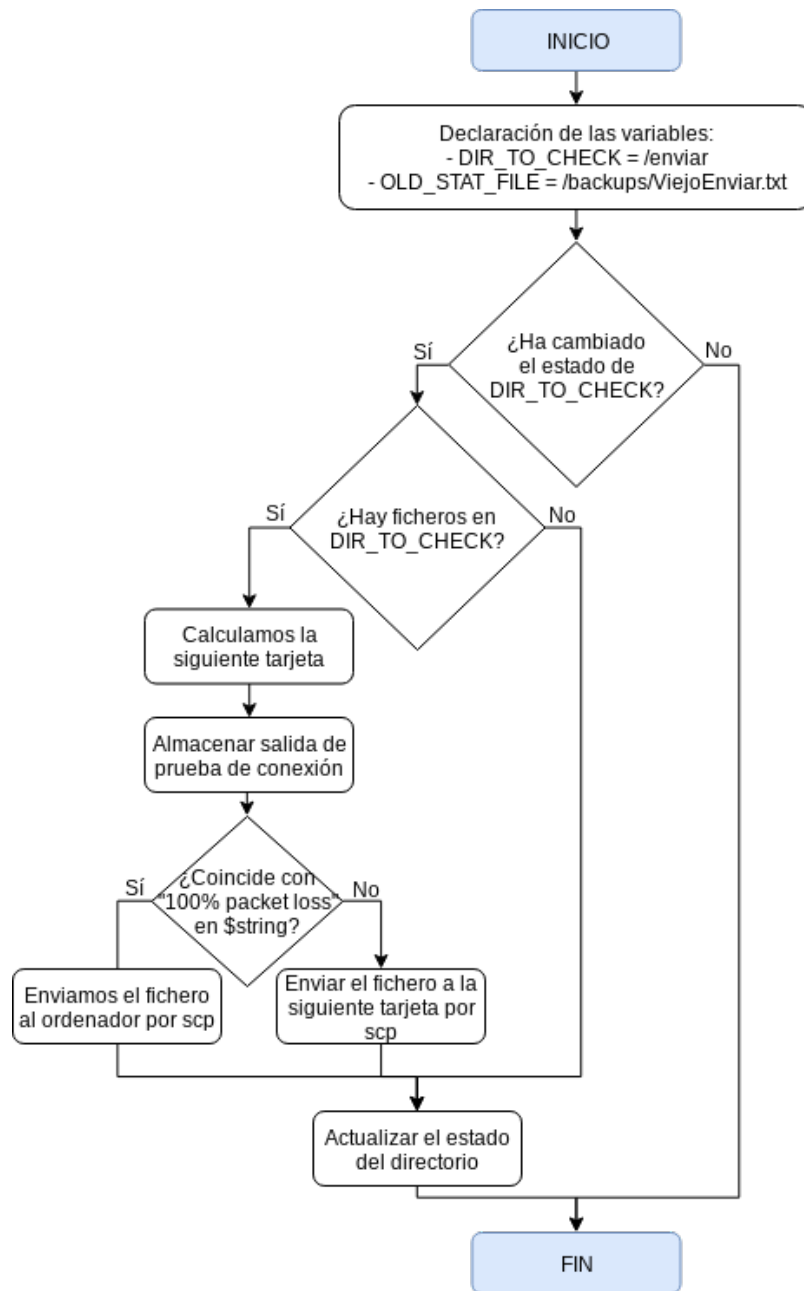


Figura B.6: Diagrama de flujo de Diagramas/Enviando.sh.

## B.6.2 Código

```

1  #!/bin/bash
2
3  #Enviando.sh
4
5  DIR_TO_CHECK=$PWD/Enviar #Directorio que queremos inspeccionar
6
7  OLD_STAT_FILE=$PWD/backups/ViejoEnviar.txt #Estado del directorio
8
9  if [ -e $OLD_STAT_FILE ]
10 then
11     OLD_STAT=`cat $OLD_STAT_FILE`
12 else
13     OLD_STAT="nothing"
14 fi
15
16 NEW_STAT=`stat -t $DIR_TO_CHECK`
17
18 if [ "$OLD_STAT" != "$NEW_STAT" ]
19 then
20
21     cd $DIR_TO_CHECK
22
23     n=$(ls | wc -l)
24     cero=0
25
26     if [[ n -ne cero ]]
27     then
28         fichero=$(ls -t | head -1) #Obtenemos el fichero más reciente
29
30         actual=$(cat /etc/passwd | cut -d : -f1 | grep zybo)
31         nActual=$(echo ${actual##*o})
32         let nSiguiente=nActual+1
33         siguiente=zybo$nSiguiente
34
35         string=$(ping -c 3 $siguiente)
36
37         if [[ $string == *100%\ packet\ loss* ]]
38         then
39             sshpass -p zybomonitor scp -o StrictHostKeyChecking=no $fichero
40                 zybo@monitor:/home/zybo/Documento/Zybo
41         else
42             siguienteZybo=$siguiente
43             siguienteZybo+=@
44             siguienteZybo+=$siguiente
45             siguienteZybo+=:/home/
46             siguienteZybo+=$siguiente
47             siguienteZybo+=/ficheros/recibir
48
49             sshpass -p $siguiente scp -o StrictHostKeyChecking=no $fichero
50                 $siguienteZybo
51         fi
52     fi
53 fi
54 echo $NEW_STAT > $OLD_STAT_FILE #Actualiza el estado del directorio
55 fi

```

Código de Enviando.sh.

## B.7 Borrar.sh

Este script se encarga de borrar el contenido de los directorios ~/ficheros/recibir, ~/ficheros/desencriptar, ~/ficheros/trabajar y ~/ficheros/enviar de las tarjetas Zybo.

Para ejecutarlo solo debemos usar el siguiente comando en el directorio ~/ficheros de las tarjetas Zybo:

```
./Borrar.sh
```

### B.7.1 Diagrama de flujo

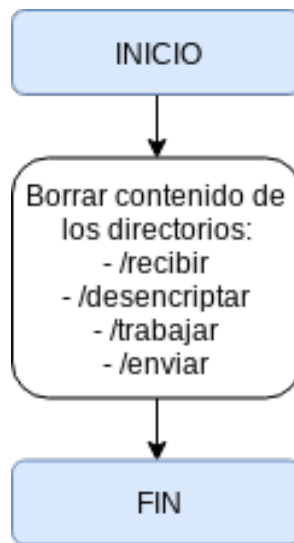


Figura B.7: Diagrama de flujo de Borrar.sh.

### B.7.2 Código

```
1 #!/bin/bash
2
3 #Borrar.sh
4
5 rm recibir/*
6 rm desencriptar/*
7 rm trabajar/*
8 rm enviar/*
```

Código de Borrar.sh.