

**INSTITUTO POLITÉCNCO NACIONAL**

Unidad Profesional Interdisciplinaria en

Ingeniería y Tecnologías Avanzadas

UNIDAD DE APRENDIZAJE:

**SISTEMAS OPERATIVOS EN TIEMPO REAL**

TRABAJO:

***REPORTE DE LA CREACION DE UNA ALARMA USANDO EL SISTEMA OPERATIVO EN TIEMPO REAL “MaRTE OS”***

ALUMNO:

* **Ruiz Hernandez David Israel**

GRUPO:

**3MV11**

PROFESOR:

**Ing. LAMBERTO MAZA CASAS**

**FECHA DE ENTREGA: Jueves 28 de Noviembre del 2019**

Contenido

[A. Introducción. 3](#_Toc26049051)

[B. Desarrollo 4](#_Toc26049052)

[C. Referencias Electrónicas. 13](#_Toc26049053)

# Introducción.

MarteOS es un sistema operativo de tiempo real para aplicaciones embebidas que sigue a la mínima en tiempo real POSIX.13. El proyecto se desarrolla por el Grupo de computación y Tiempo Real en la Universidad de Cantabria, aunque también existen colaboradores en distintos lugares.

El entorno de desarrollo se basa en la GNU compiladores GNAT, GCC y GCJ. La mayor parte de su código está escrito en Ada con algunas partes C y ensamblador.

**Características**

Entre las características principales del kernel Marte OS se tiene:

* Soporta aplicaciones de lenguaje mixto en Ada, C y C++ (soporte experimental para Java también).
* Ofrece los servicios definidos en POSIX.13: pthreads, exclusiones mutuas, condvars.
* Todos los servicios tienen una respuesta de tiempo limitado (incluida la asignación de memoria dinámica con TLSF ).
* Espacio de direcciones de memoria individual compartida por la aplicación multi-hilo y Marte OS.
* Disponible bajo la Licencia Pública General GNU 2.
* Sobre la base de la cadena de herramientas GNU AdaCore.
* Implementa el anexo Ada2005 Tiempo Real

**Contribuciones por usuarios**

Algunas contribuciones por usuarios de Marte OS

* Protocolos de comunicación y middleware
* PolyORB (DSA y CORBA) y GLADE para MaRTE OS x86 para equipos sin Sistema Operativo con RT-EP
* RT-WMP Real-Time Wireless protocol (UNIZAR)
* Protocolo RT-EP con una capa de ancho de banda reservado, algoritmos de exclusión mutua (Mutex) distribuidos, servicios de broadcast
* FRESCAN protocolo de red para bus CAN con ancho de banda reservado y servidores esporádicos
* DTM, el Gestor de Transacciones Distribuidas de FRESCOR
* MyCCM, OMG CCM (THALES)
* Drivers
* CAN bus
* Wireless ralink rt61
* SVGA, BTTV, Soundblaster 16
* Mouse, Keyboard, Joystick
* Ethernet drivers (intel eepro100, rtl8139, SiS900)
* Serial port driver
* I2C protocol, compass CMPS03 driver
* IDE disk driver (CompactFlash and HD) and FAT 16 filesystem

# Desarrollo

Teniendo instalado el sistema operativo MaRTE OS dentro de la aplicación de DEBIAN para WINDOWS nos dirigimos a la siguiente dirección usando el comando *cd* de la siguiente forma: *“cd /myapps/marte\_2.0\_22Feb2017/examples”*, ahora tecleamos el comando *vim* para crear un nuevo archivo con el nombre que nosotros deseemos para colocar el código: “vim segundos\_con\_scanf.c”.

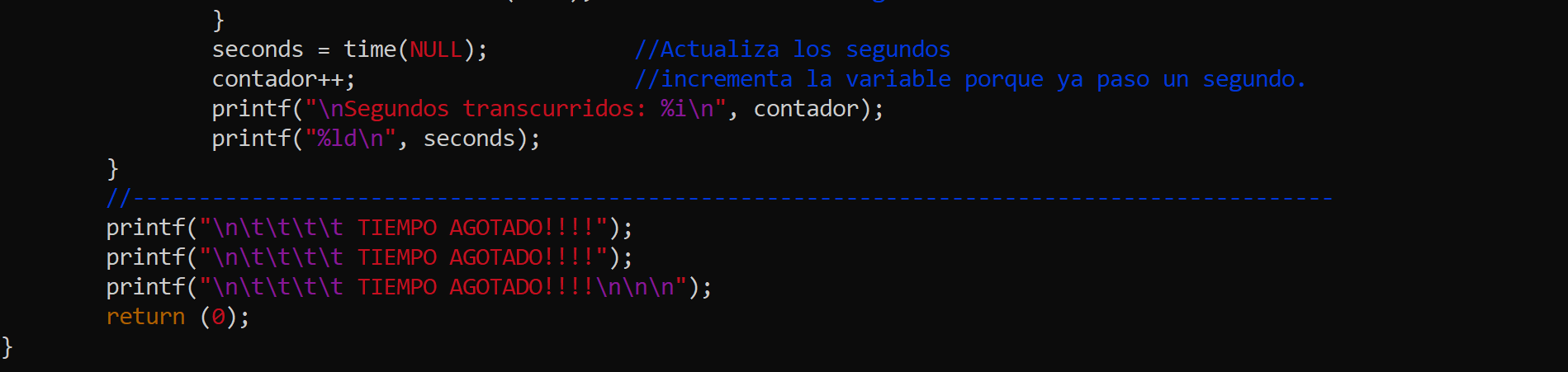
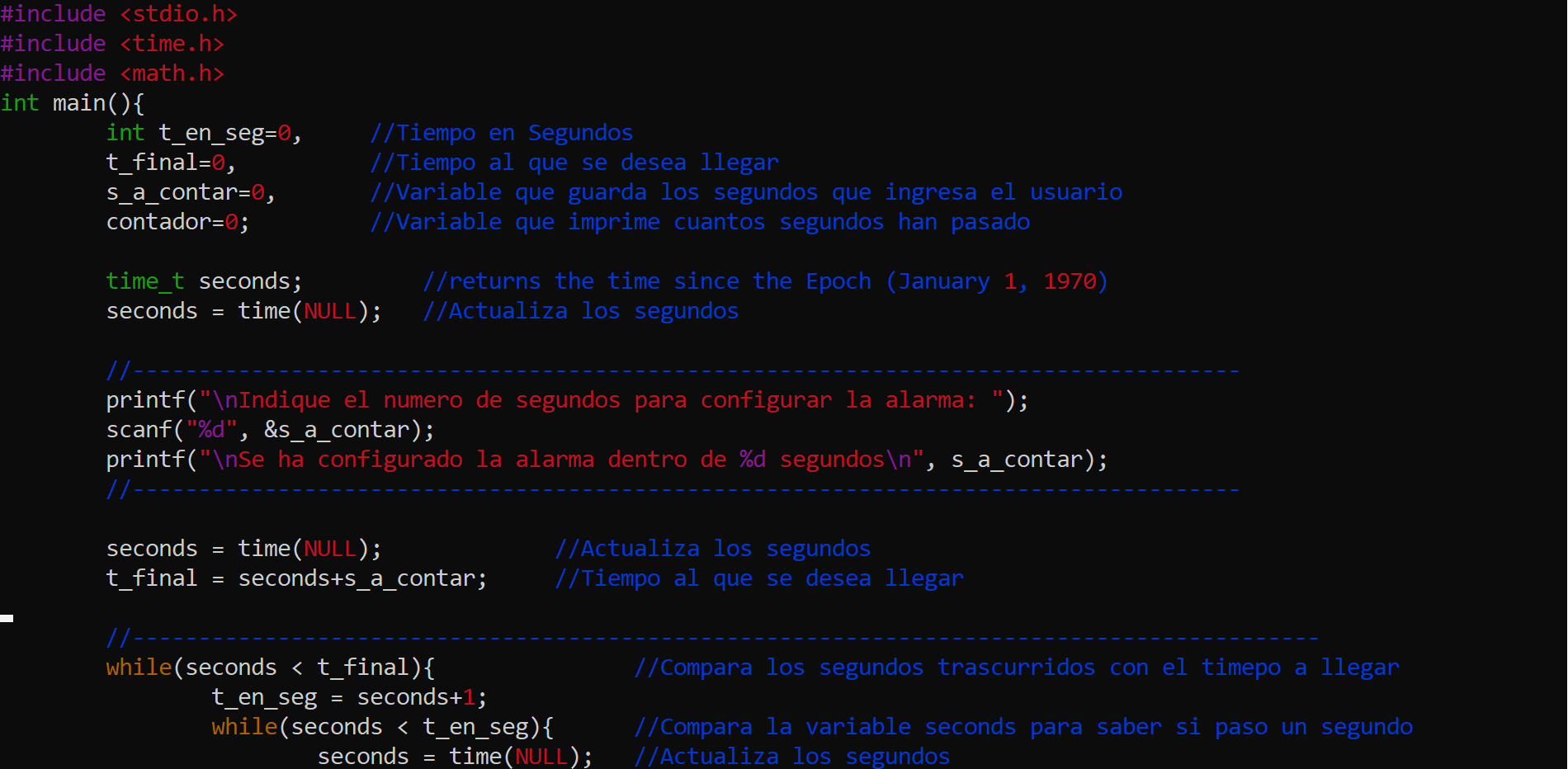


Ilustración 1 Código de la Alarma

Al terminar de escribir el código para la Alarma presionamos la tecla “*Esc”* para salir del modo “*Insert”*, para después teclear *“:x”* para salir del Editor *“vim”* y guardar el archivo. Ahora usaremos el siguiente comando *“mgcc segundos\_con\_scanf.c”* para compilar el archivo que acabamos de crear, para verificar que la compilación fue exitosa se debe crear un archivo con el nombre *“a.out”*, para verificar la existencia del archivo usamos el comando *“ls”*.

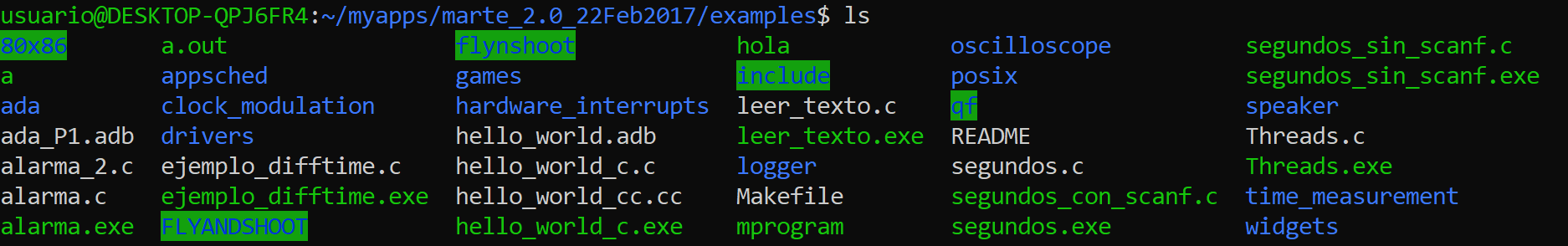


Ilustración 2 Verificación de la creación del archivo "a.out". usando el comando “ls”.

Ahora ejecutamos la instrucción *“mgcc segundos\_con\_scanf.c -o mprogram”* para crear un archivo de nombre “mprogram” que esta asociado a la compilación del archivo “*segundos\_con\_scanf.c”*, para verificar que el archivo se creó correctamente usamos el comando *“ls”*.

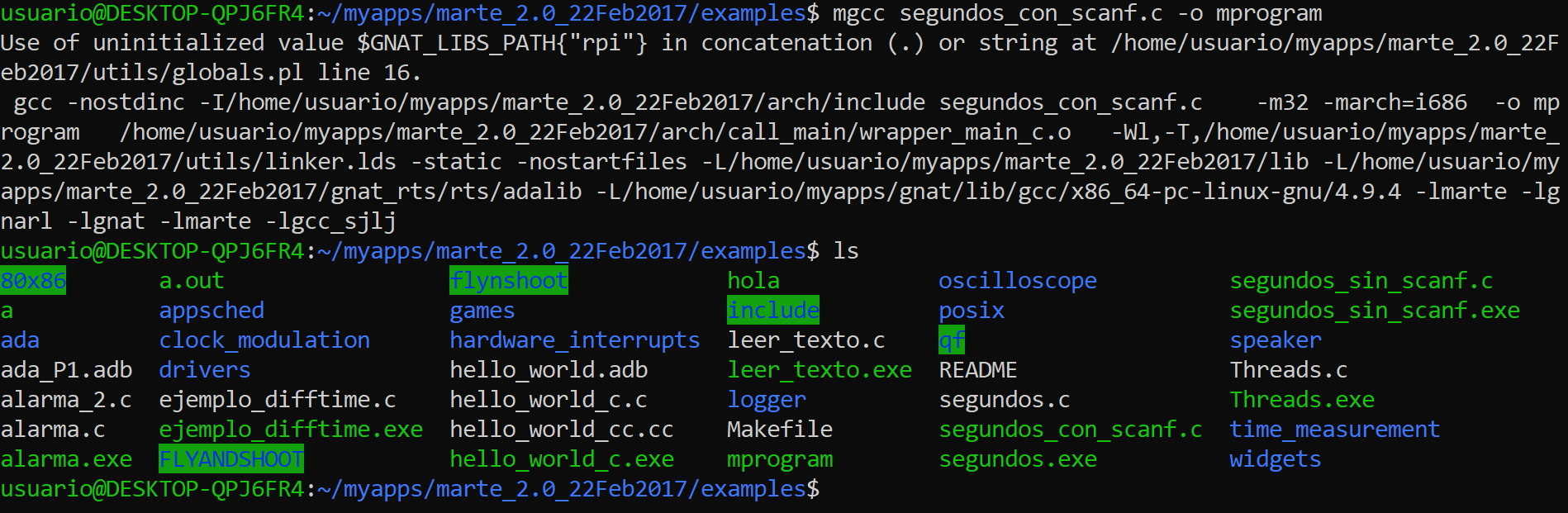
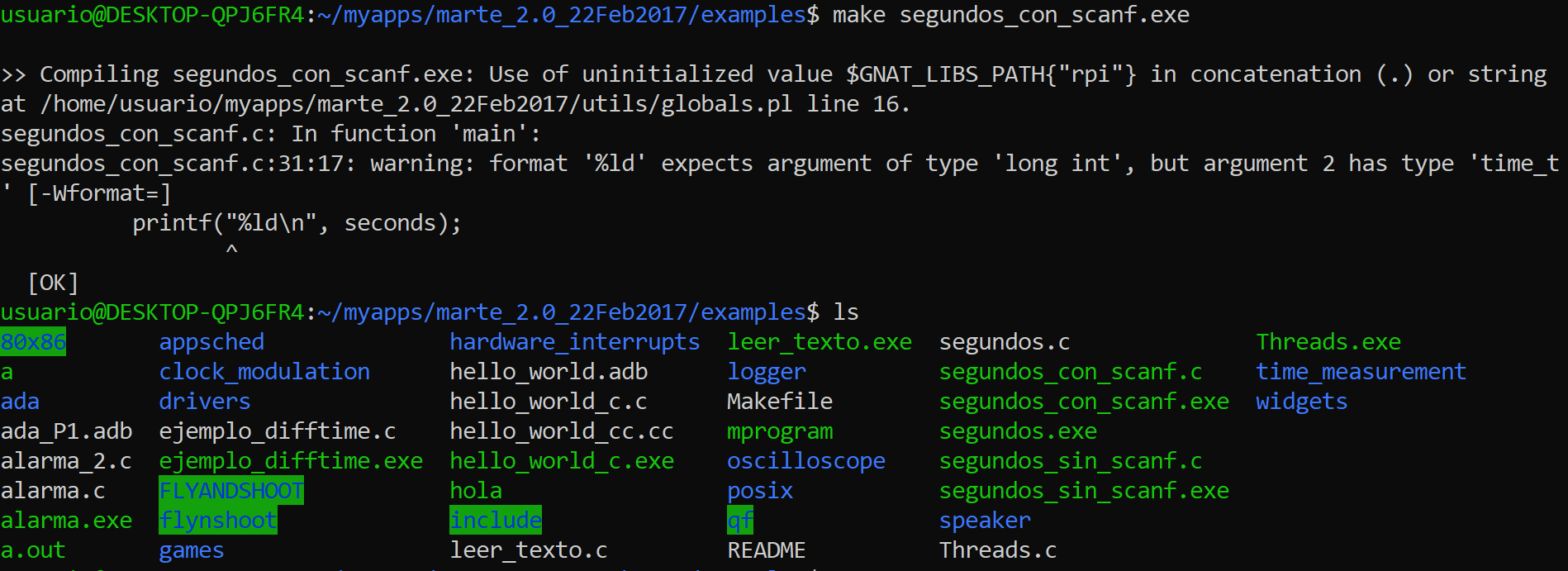


Ilustración 3 Creación del archivo "mprogram" usando la instrucción “mgcc segundos\_con\_scanf -o mprogram”.

El siguiente paso es ejecutar la instrucción *“make segundos\_con\_scanf.exe”*, con esto crearemos un archivo ejecutable asociado con el archivo C, con el comando *“ls”* corroboramos que se halla creado el archivo.

Ilustración 4 Creación del archivo "segundos\_con\_scanf.exe"



Para simular el programa que se acaba de crear se debe instalar la aplicación “Xming” de la siguiente dirección:

<https://sourceforge.net/projects/xming/>



Ilustración 5 Icono de la aplicación "Xming".

Después de que termina de instalar la aplicación debemos ejecutarla, esta lo hará en segundo plano por lo que solamente veremos el icono en la “Barra de iconos”. Ahora dentro de la aplicación “Debian” tecleamos la siguiente instrucción “qemu-system-i386 -kernel segundos\_con\_scanf.exe”.

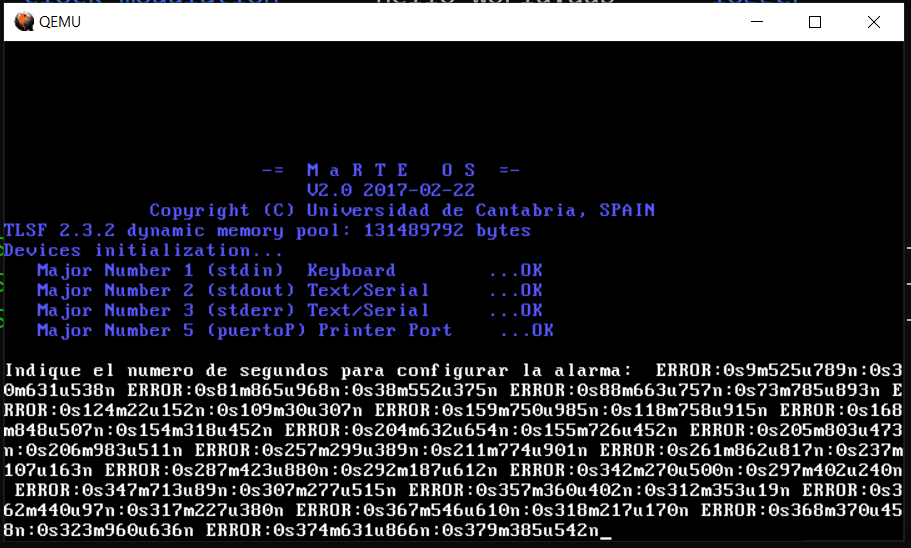


Ilustración 6 Ejecución del programa "Segundos\_con\_scanf.exe" usando la aplicación "Xming".

Como se puede observar en la “Ilustración 6” se presentan algunos errores que no dejan que el programa se ejecute correctamente, estos suelen desaparecer cuando se crea la Live USB.

Lo siguiente es crear la Live USB usando la aplicación “Lili USB Creator” el cual se debe descargar del siguiente link:

<http://www.linuxliveusb.com/en/download>



Ilustración 7 Icono de la aplicación Linux Live USB Creator.

Además, debemos descargar y descomprimir en el escritorio el siguiente link, donde tenemos el “kernel.elf” y el “Makefile” para poder crear la USB.

<https://github.com/sotrteacher/sotr_201808_201812/tree/master/PRACTICA_1_Instalacion_de_un_SOTR>

Al término de los pasos anteriores, nos dirigimos a Debian y copiamos el archivo “segundos\_con\_scanf.exe” al siguiente directorio “C:\Users\Isra\_el\_\Desktop\sotr\_201808\_201812-master\PRACTICA\_1\_Instalacion\_de\_un \_SOTR\MAKEFILE\_GRUB\_Legacy\_Kernel\_elf” usando el comando “cp -v”.

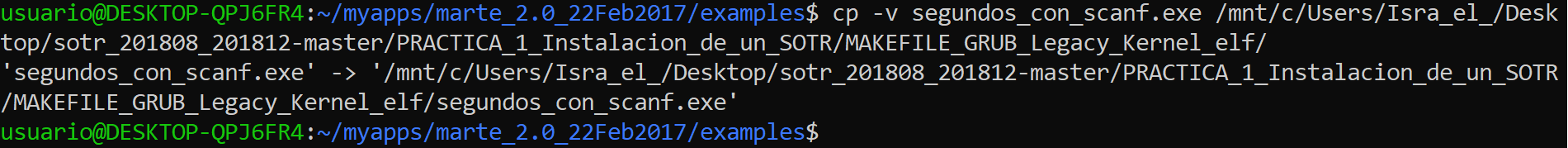


Ilustración 8 Uso del comando "cp -r" para copiar archivos a un directorio.

Ahora necesitamos acceder a ese directorio usando el comando “cd”, para después el archivo .exe al “kernel.elf” usando el comando “cp -v”.

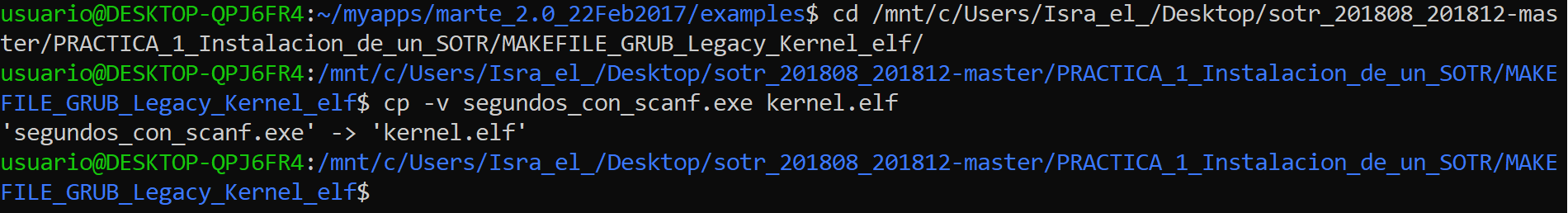


Ilustración 9 Creación del archivo "kernel.elf" usando el archivo "segundos\_con\_scanf".

Por último, debemos crear el archivo “ISO” usando el comando “make”.

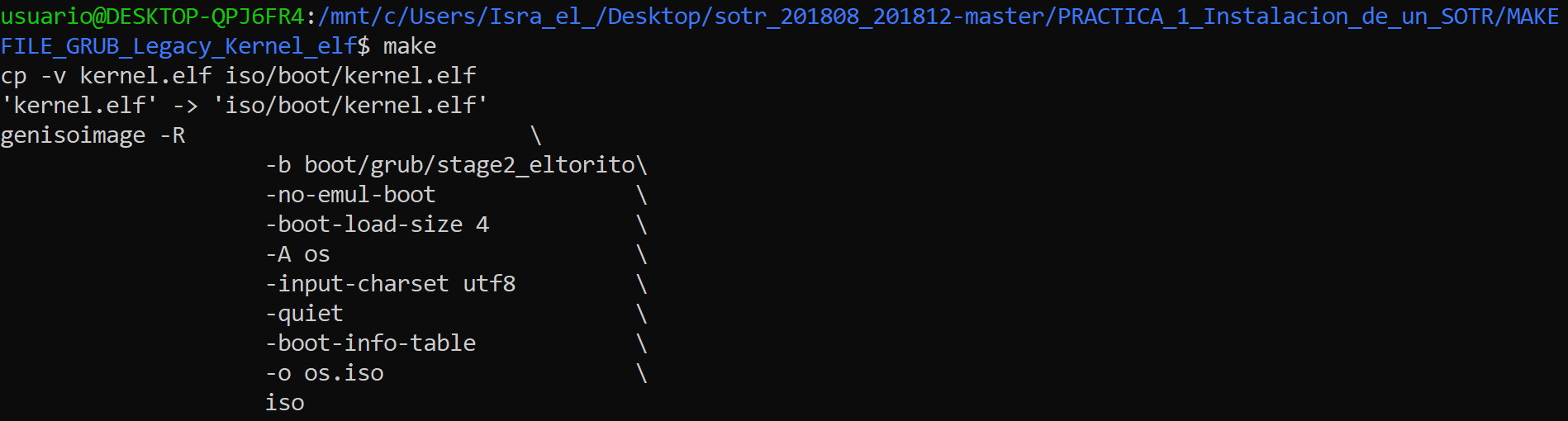


Ilustración 10 Creación del archivo "ISO" usando el comando "make".

Para crear la Live USB debemos conectar una usb a la computadora y ejecutar el programa “Linux Live USB Creator”, donde debemos elegir la unidad que acabamos de conectar y el archivo ISO que acabamos de crear, por recomendación se debe seleccionar la pestaña de “Formatear Dispositivo”, por ultimo debemos hacer clic en el rayo para comenzar la instalación.



Ilustración 11 Programa para crear la Live USB.

El siguiente paso es reiniciar la computadora y presionar la tecla “F12” para entrar a la configuración de arranque y poner en primer lugar el puerto usb. De esta forma podemos ver como se empieza a ejecutar nuestro programa sobre el sistema operativo tiempo real MaRTE OS.

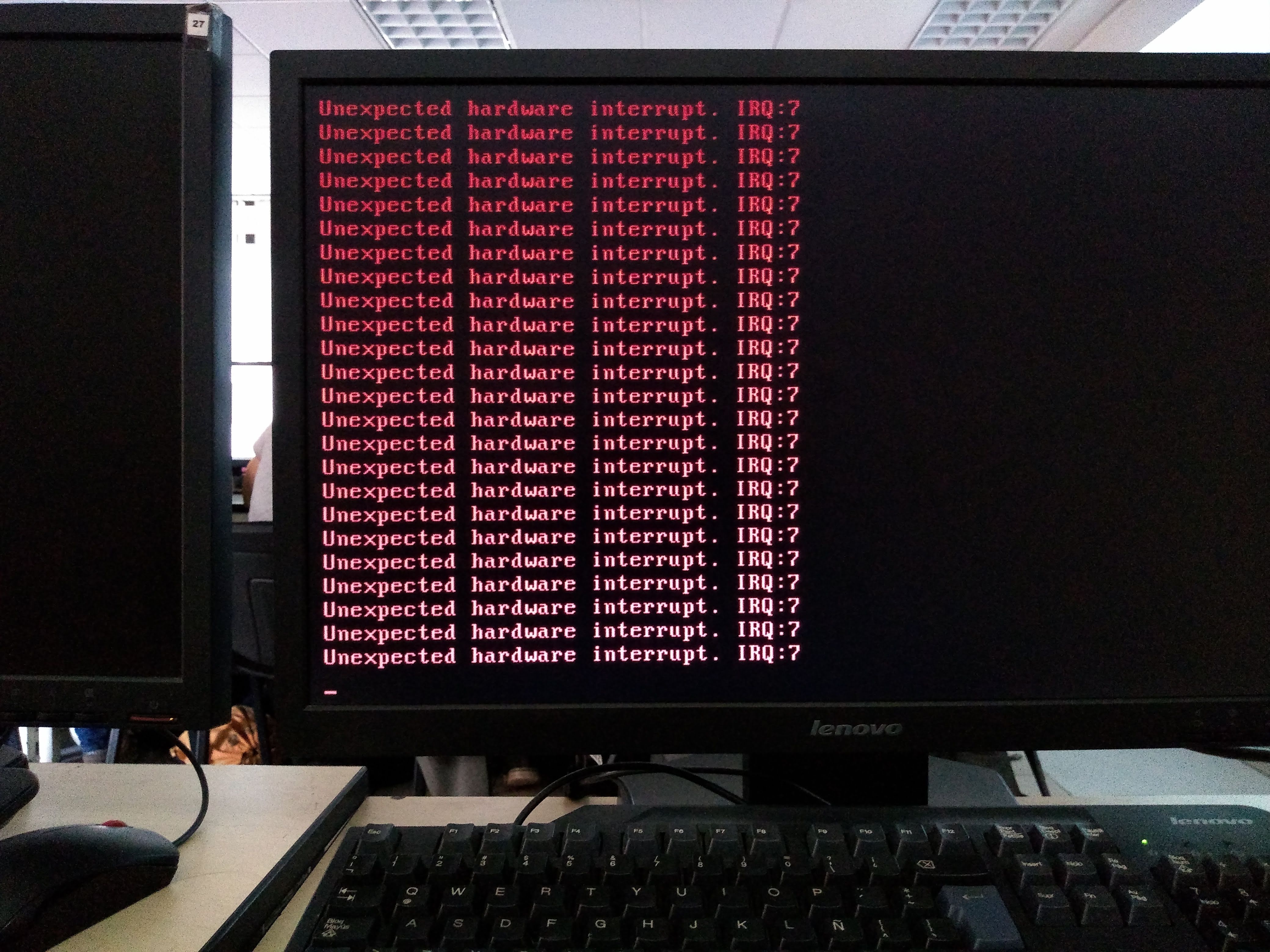


Ilustración 12 Se muestran los errores que aparecen al intentar ejecutar el programa “segundos\_con\_scanf”.

Como se puede observar en el “Ilustración 12” al ejecutar el programa aparece el error: *“Unexpected hardware interrup. IRQ:7”*. Para evitar que este error apareciera en la pantalla se decidió comentar esas líneas. Usando la instrucción *“grep -r ‘Unexpected hardware’ /home/usuario/myapps/marte\_2.0\_22Feb2017”.*

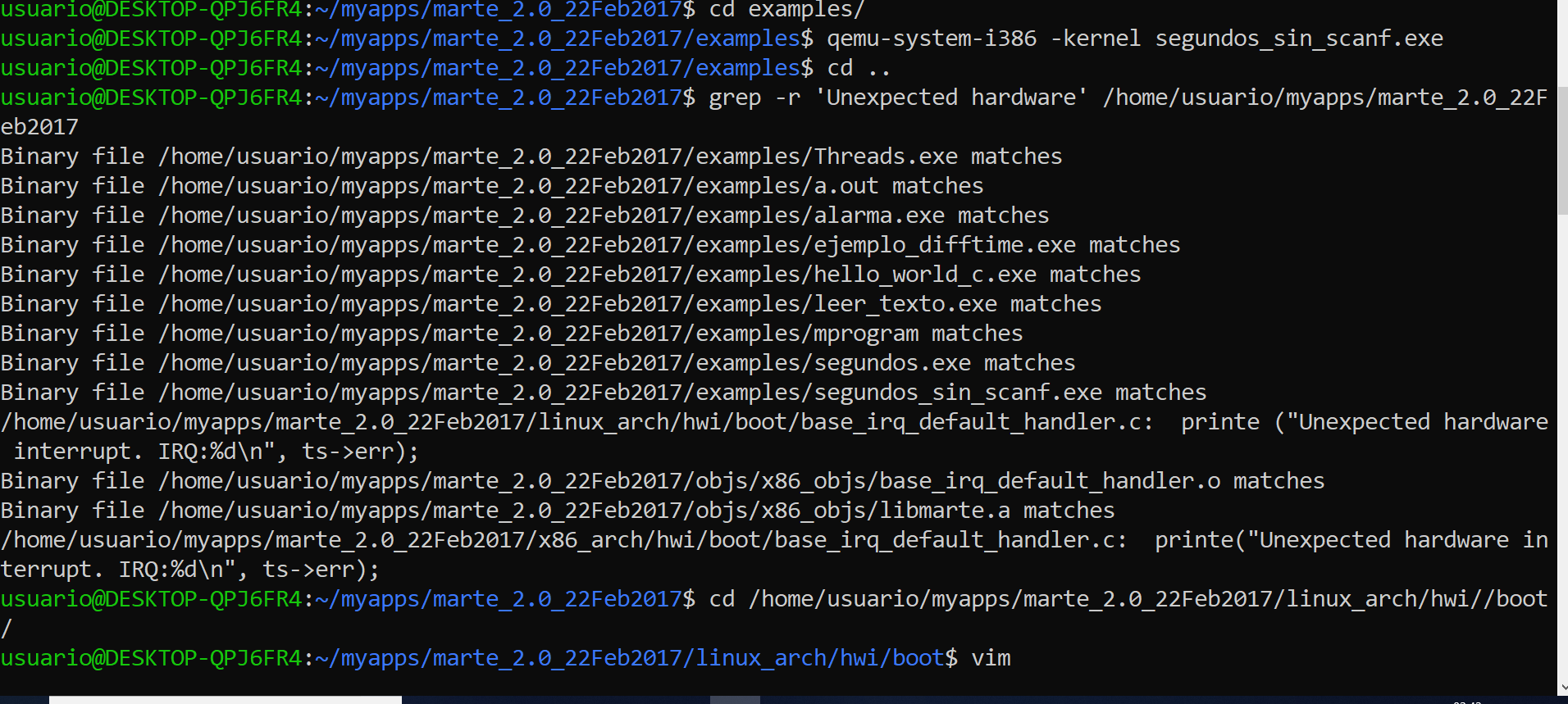


Ilustración 13 Uso del comando "grep -r" para buscar palabras dentro de documentos.

De acuerdo a la “Ilustración 13” tenemos 2 archivos que debemos modificar y comentar las líneas donde aparece *“Unexpected hardware interrup. IRQ:7”*. El primero esta dentro de la carpeta *“linux\_arch”* y el segundo *“x86\_arch”* y para ambos el nombre del archivo es *“base\_irq\_default\_hander.c”*.

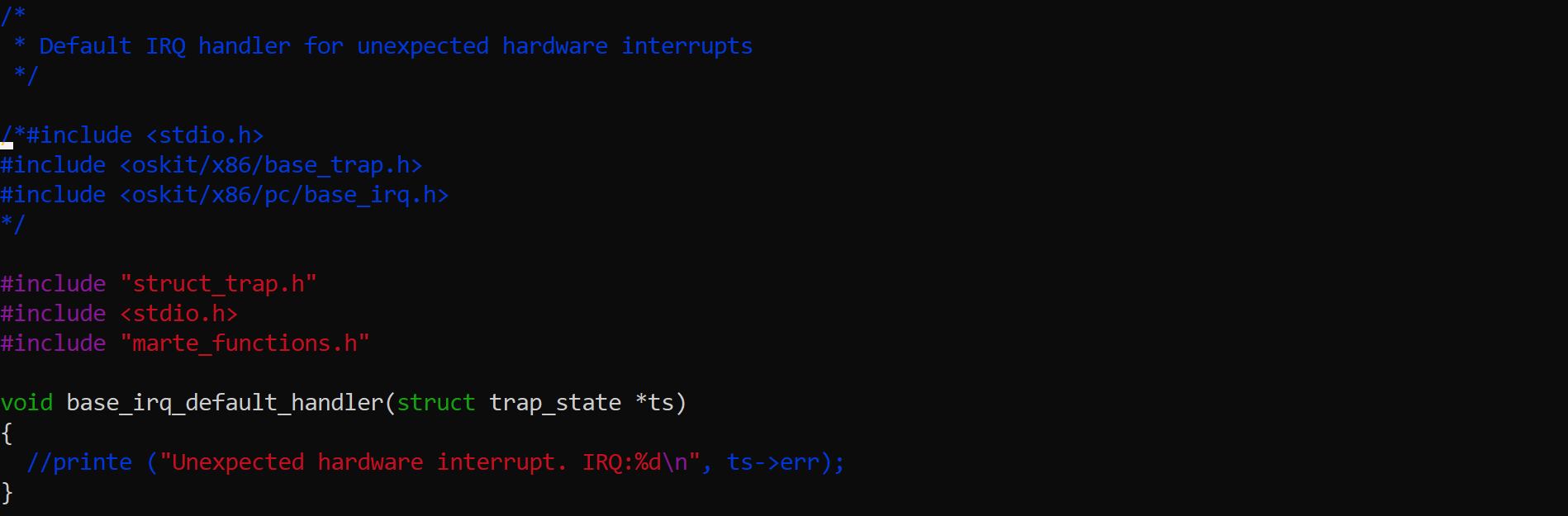


Ilustración 14 Modificación del archivo "base\_irq\_default\_handler.c".

Ahora debemos usar el comando “mkmarte” para volver a compilar el sistema operativo en tiempo real MaRTE OS y que los cambias que acabamos de hacer se apliquen, si el proceso fue exitoso debe aparecer un mensaje como el de la *“Ilustración 15”.*

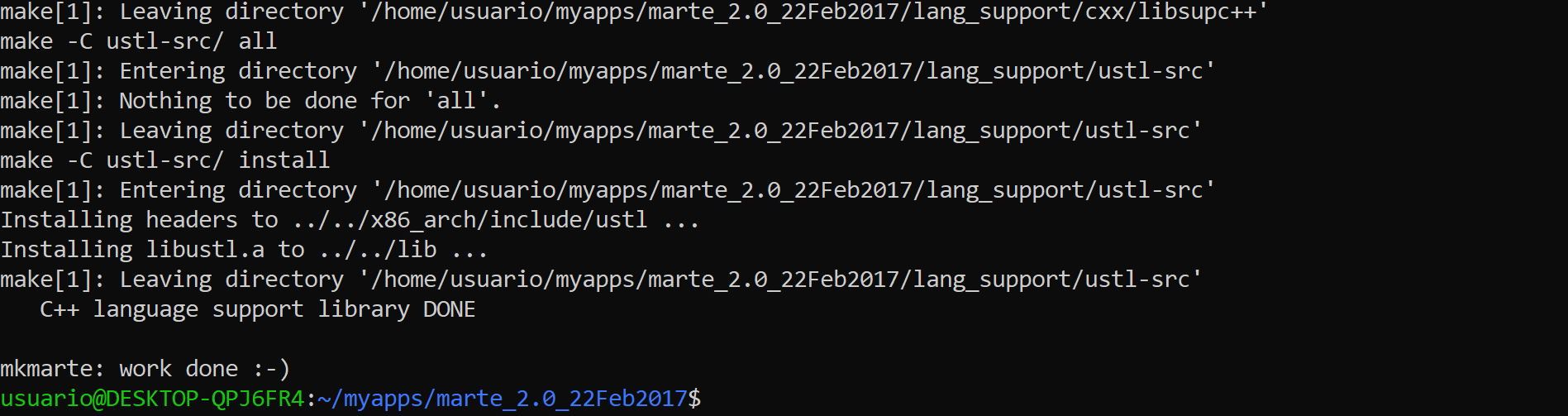


Ilustración 15 Mensaje "work done" que aparece al termino de usar el comando "mkmarte".

Al término de la compilación de MaRTE, debemos de volver a realizar los pasos desde que usamos la instrucción *“mgcc segundos\_con\_ scanf.c”* para crear el archivo .exe con los cambios que se realizaron al sistema operativo en tiempo real. También debemos de crear la LIVE USB con el nuevo archivo *“segundos\_con\_scanf.exe”*. Al reiniciar nuestra computadora y tener conectada la memoria con el nuevo archivo .iso nuestro programa de la ALARMA se debe ejecutar correctamente.

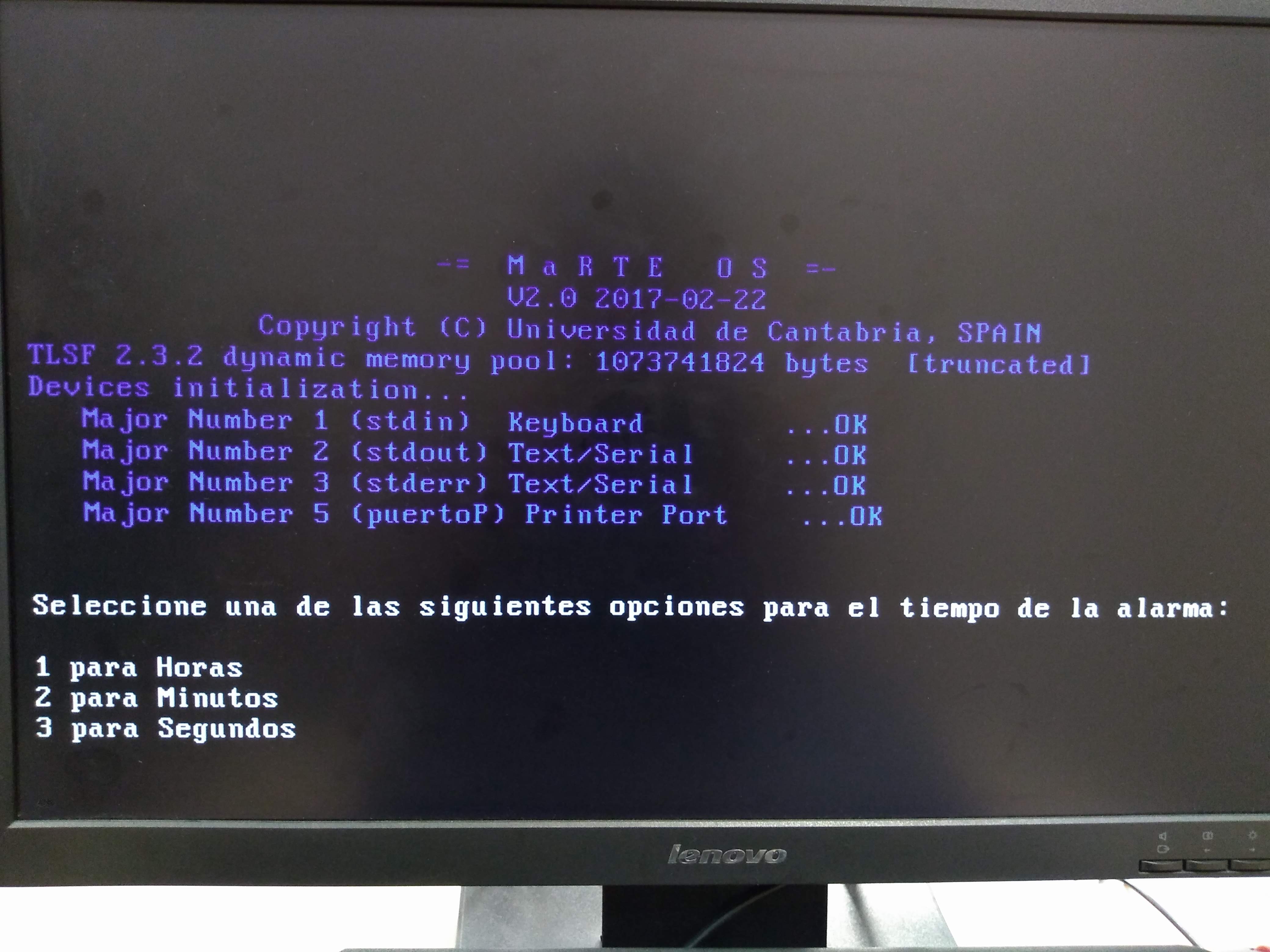


Ilustración 16 Ejecución del programa "segundos\_con\_scanf.c". usando MaRTE OS.

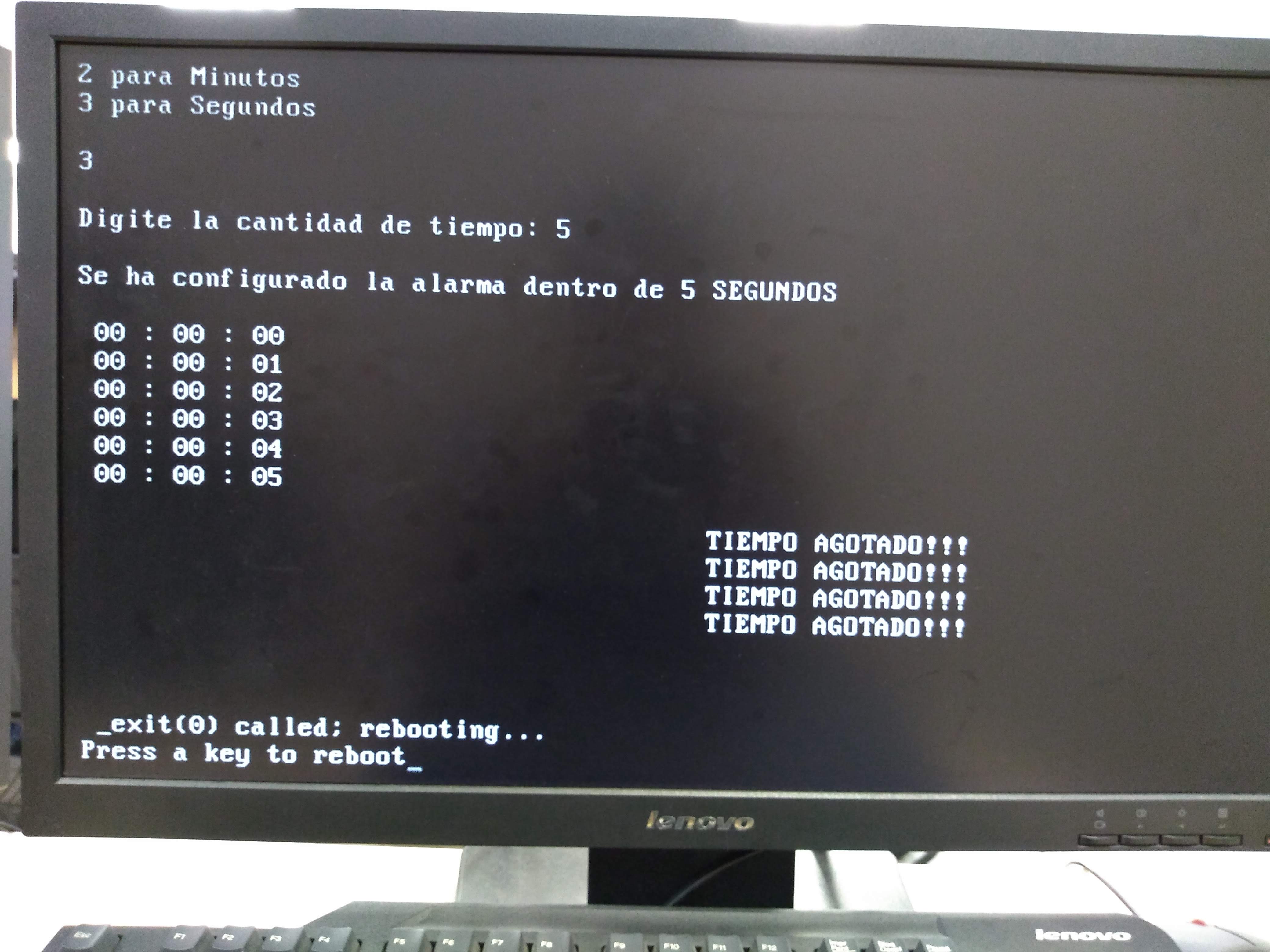


Ilustración 17 Configuración de la alarma usando MaRTE OS.

Observando la *“Ilustracion 16”* nos damos cuenta que se pudo ejecutar correctamente el programa *“segundos\_con\_scanf.c”* sin que se mostrara ningún error en la pantalla. En *la “Ilustración 17”* se configura la alarman dentro de 5 SEGUNDOS, pasando el tiempo se muestra en pantalla un mensaje *de “TIEMPO AGOTADO”* lo que nos demuestra que nuestra alarma funciona correctamente.

# Referencias Electrónicas.

<https://es.wikipedia.org/wiki/MaRTE_OS>

<https://marte.unican.es/>

<https://www.ctr.unican.es/asignaturas/MC_ProCon/Doc/ProCon_II_02-marte_3en1.pdf>

<https://prezi.com/uklrnt37zeg6/marte-os/>