UNIVERSIDAD NACIONAL DE CÓRDOBA

Facultad de Ciencias Exactas, Físicas y Naturales

Cátedra de Sistemas Operativos II

Profesor: Ing. Agustín Martina

TRABAJO PRÁCTICO 3

Programación con CGI

Mauricio Aguilar

29 de Mayo de 2016

Contenido

Introducción

Interfaz de entrada común (en inglés Common Gateway Interface, abreviado CGI) es una importante tecnología de la World Wide Web que permite a un cliente (navegador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Es un mecanismo de comunicación entre el servidor web y una aplicación externa cuyo resultado final de la ejecución son objetos MIME. Las aplicaciones que se ejecutan en el servidor reciben el nombre de CGIs.

Las aplicaciones CGI fueron una de las primeras prácticas de crear contenido dinámico para las páginas web. En una aplicación CGI, el servidor web pasa las solicitudes del cliente a un programa externo. Este programa puede estar escrito en cualquier lenguaje que soporte el servidor, aunque por razones de portabilidad se suelen usar lenguajes de script. La salida de dicho programa es enviada al cliente en lugar del archivo estático tradicional.

CGI ha hecho posible la implementación de funciones nuevas y variadas en las páginas web, de tal manera que esta interfaz rápidamente se volvió un estándar, siendo implementada en todo tipo de servidores web.

En el presente trabajo utilizaremos CGI para mejorar las características de la AWS del trabajo práctico 1.

Descripción del problema

Nuevamente el Servicio Meteorológico Nacional le solicita a la Escuela de Ingeniería en Computación de la FCEFyN-UNC, que realice el diseño, desarrollo y testing del sistema de adquisición de datos de Estaciones Meteorológicas Automáticas (AWS), en su segunda versión, utilizando nuevamente la plataforma de desarrollo Intel Galileo.
Se solicita que, a partir de lo realizado en el Trabajo Practico Nº1, se desarrolle lo siguiente:
1) Que se sincronice el registro intermedio entre el Simulador de generación de datos y el proceso de la
AWS, utilizando la API de sincronización del sistema operativo (se sugiere utilizar semáforos).
2) Que se realice un estudio de los distintos web servers disponibles para sistemas embebidos, realice una comparación y justifique la selección de uno de ellos, que deberá instalar en el sistema operativo, y que deberá ejecutarse automáticamente cada vez que este se reinicia.
3) Sobre el servidor web, debe desarrollarse una interfaz web simple (NO PHP!), con múltiples pestañas, donde cada pestaña debe mostrar, utilizando un programa CGI, lo siguiente:
a) Reporte información sobre recursos varios del sistema embebido (procesador, memoria, uptime, etc.).
b) Pestaña que permita ejecutar los comando desarrollados para el TP1, sin la utilización de sockets y con las siguientes modificaciones:
• get_telemetry: abre en una nueva ventana la última telemetría
• get_datta: abre el total de telemetrías obtenidas en una nueva ventana
• erase_datta: vacía el buffer de datos obtenidos
• NO implementa connect y disconnect (dado que no estamos trabajando con sockets)
b) Pestaña que muestre el listado de módulos instalados
c) Formulario que permita subir un archivo al servidor, controlar que este sea un archivo válido (del tipo módulo), e instalarlo en el kernel del sistema operativo.
También debe poseer un botón para removerlo.
4) Desarrollar un módulo (driver) simple y vacío, que sólo imprima “Hello World” al instalarse y “Good Bye World” al ser removido del kernel. Este será el módulo que se debe instalar en la pestaña c del punto 3.

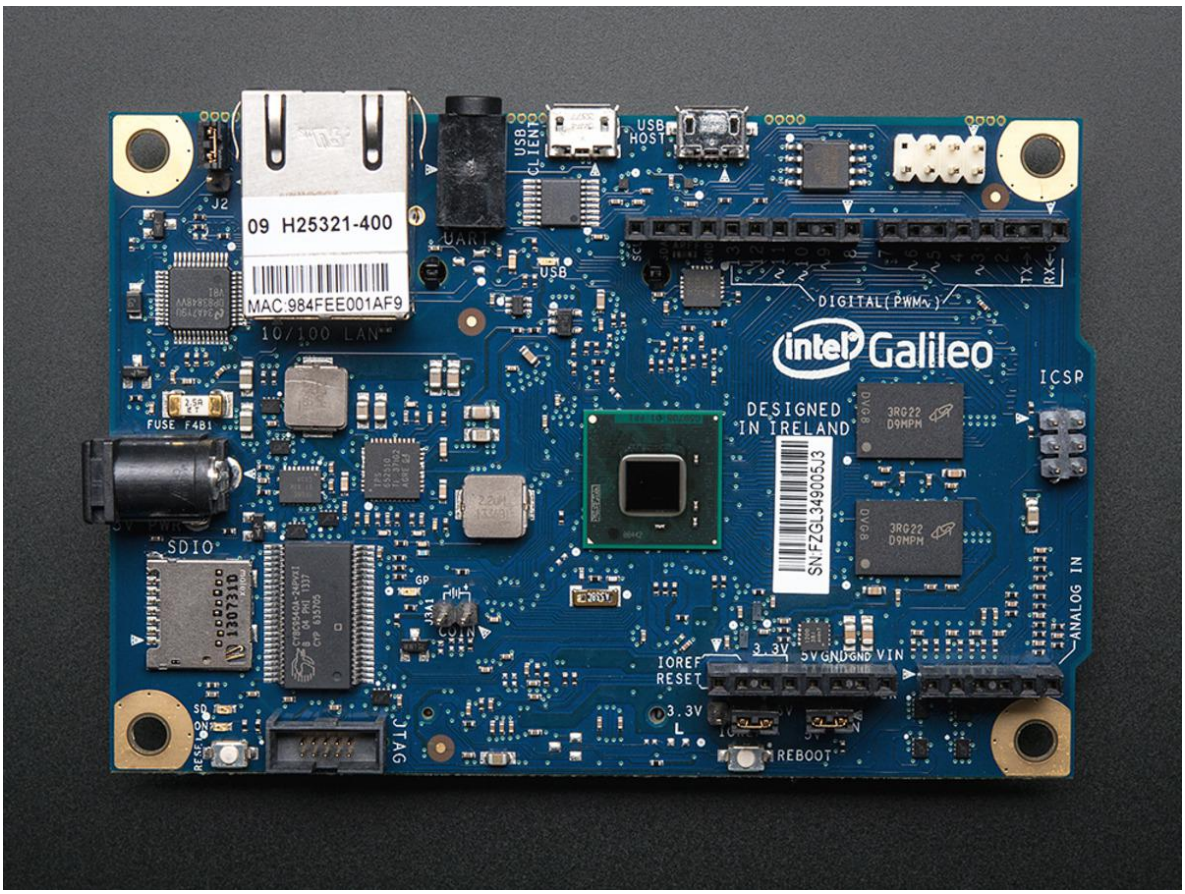
Requerimientos y tareas

Requerimientos Funcionales:

- El usuario accedera por navegador a la AWS introduciendo su IP.
- El usuario podrá seleccionar una de las siguiente opciones del menú principal:
  - Obtener información del sistema
  - Get_Telemetry
  - Get_Datta
  - Erase_Datta
  - Ver modulos instalados
  - Subir modulos
  - Instalar modulo
  - Quitar modulo
- Obtener información: El usuario recibirá en pantalla información del sistema
- Get_Telemetry: El usuario recibirá el ultimo dato sensado.
- Get_Datta: El usuario recibirá todos los datos sensados.
- Erase_Datta: El usuario borrará todos los datos sensados.
- Ver modulos: El usuario verá todos los módulos que corren en la Galileo.
- Subir modulo: El usuario enviará un modulo a la Galileo.
- Instalar modulo: El usuario instalara el modulo enviado en la Galileo.
- Quitar modulo: El usuario desintalara el modulo de la Gelileo.

Diseño y documentación

Para la placa Galileo Intel I que se muestra en la siguiente figura, existen al menos dos distribuciones conocidas para utilizar. La imagen Yocto oficial de Intel y un Debian Wheezy.



Se eligió la imagen oficial de Intel porque luego de muchos intentos con Debian Wheezy y su servidor lighttpd, nose logró terminar de configurar. La desventaja de usar la imagen oficial es tener que hacer un cross-compiling para el módulo.

En cuanto a los servidores webs, se han investigado algunos como thttpd, ligthttpd (que viene por defecto en la imagen de Intel pero sin el modulo CGI activo), Xampp, etc.

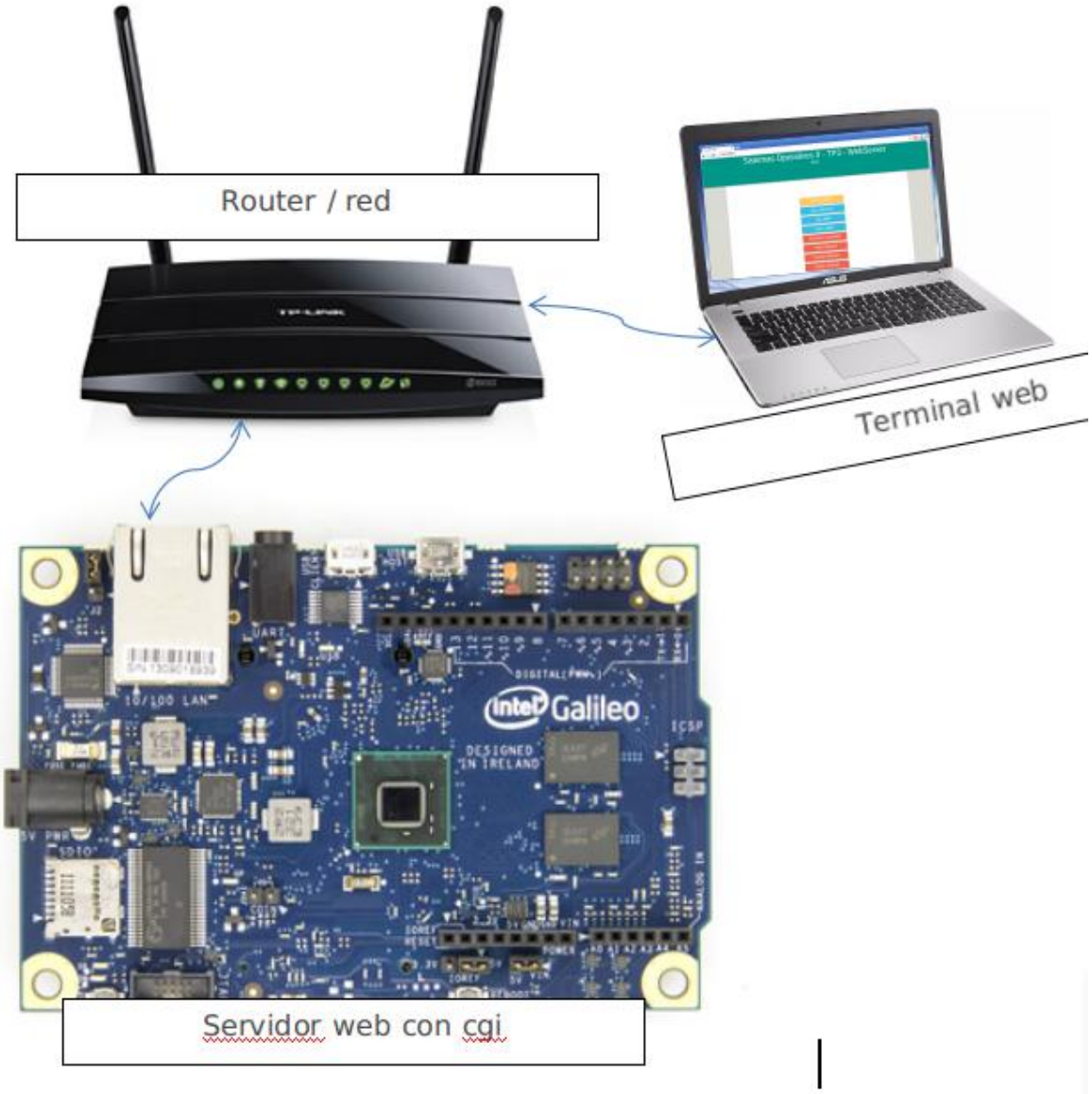
Se eligió thttpd por su pequeño tamaño de unos pocos MB. Si bien lighttpd es liviano, es algo complicado activar cgi. Xampp por su parte no es un servidor apto ya que no solo ocupa mucho espacio, sino que además su instalador es gráfico.

El sitio oficial de thttpd es: http://www.acme.com/software/thttpd/

.

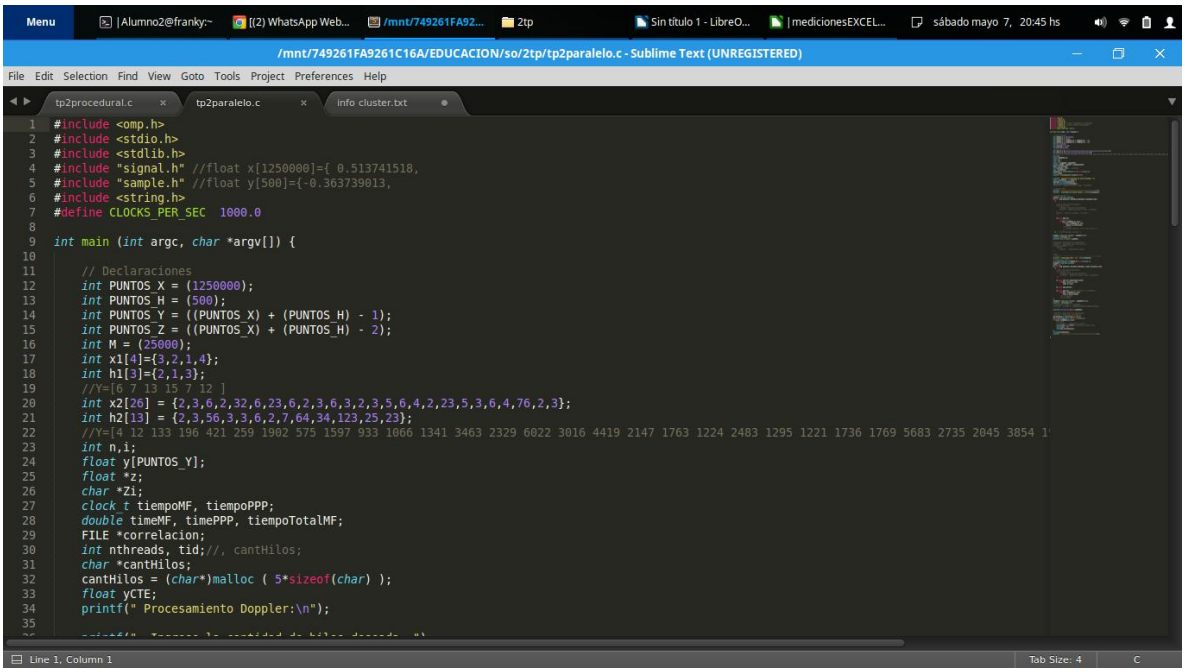
Diagrama de Despliegue

En el siguiente diagrama podemos observar la interconexion de los componentes del sistema:



Implementación y Resultados

Para codificar se utilizó la herramienta Sublime Text 3:



Para compilar, se utilizó gcc por terminal de Linux.

Proceso de montado del sistema paso a paso:

1- Descargar la imagen oficial desde:

https://software.intel.com/galileo-image/latest

2- Desde Windows, instalar RawWrite32, abrirlo, seleccionar imagen descargada en el paso 1, insertar sd a grabar en la computadora, seleccionarla en RawWrite32 (tener en cuenta su tamaño para seleccionarla correctamente), y hacer clic en Write.
Desde linux se puede hacer con el comando dd:

dd if=origen.img of=destino_SD bs=1M
ej: dd if=/dev/sda1/imagen.img of=/dev/sdb bs=1M
se debe consultar origen y destino con: fdisk -l

Si bien podríamos ampliar la partición con Gparted de linux, hasta por ejemplo, unos 2GB, esto trae problemas de funcionamiento.

3- Colocamos sd en la placa Intel Galileo I, conectamos el cable de red RJ45 del router a la placa, y finalmente la alimentación eléctrica por el cable Jack.

4- Ingresamos por ssh a la placa:

sudo ssh root@ip]
ej: **sudo ssh root@192.168.0.101**

la primera vez nos pregunta si estamos seguros, ponemos yes
y la contraseña por defecto es root

Si borramos la sd y colocamos la misma u otra imagen, nos advertirá que el sistema cambio, a modo de protección sniffing y no nos dejara entrar.
Debemos borrar el archivo:

/(usuario)/.ssh/known-hosts y volver a ingresar.

En el mensaje de advertencia figura dicho archivo.

sudo rm /(usuario)/.ssh/known-hosts

5- Una vez dentro de la placa a través de la termina, descargar servidor:

cd /tmp

wget http://acme.com/software/thttpd/thttpd-2.27.tar.gz

6- Descomprimir archivo en la carpeta /tmp

tar -zxvf thttpd-2.27.tar.gz

7- Entrar a la carpeta que contiene los archivos descomprimidos.

cd /thttpd-2.27

8- Para evitar el error de que no existe el grupo "www", agregarlo con:

comando groupadd www

9- Para evitar el error de que no existe la carpeta "/usr/local//man/man1", crearla:

cd comando mkdir /usr/local/man/man1

10- Configurar el entorno de compilación:

./configure

11- Generar instalador:

make

12- Instalar:

make install

13- Crear archivo de configuración en /etc

vi /etc/thttpd.conf

Y agregamos la linea

port=8080 dir=/var/www/ user=root cgipat=**.cgi logfile=/var/log/thttpd.log pidfile=/var/run/thttpd.pidi

14- Crear llamada al archivo de configuración en el inicio del sistema

vi /etc/init.d/thttpd

poner imagen

15- Dar permisos de ejecución al archivo de inicio mediante

chmod ugo+x /etc/init.d/thttpd

16- Actualizar el inicio del sistema:

comando update-rc.d thttpd defaults

17- Apagar sistema y volver a encender:

shutdown now

desenchufar, esperar 5 seg, enchufar

foto de proceso al inicio

18- Cuando todo funciona, apagamos el sistema, quitamos la tarjeta sd de la placa Galileo y la ponemos en alguna computadora con Linux. Procedemos a hacer una copia de la sd con el comando dd:

dd if=/dev/mmcblk0p2 of=/home/mmcblk0p2-GalileoConThttpd.bin

El origen y el destino deben determinarse con el comando fdisk -l

Configuración y utilización del servidor

Estructura de archivos del servidor:

* Los archivos HTML se guardan en /var/www/

* Los archivos CGI se guardan en /var/www/cgi-bin/

* Los archivos que se suben desde la pagina se guardan en /var/www/upload/

Asignar permisos:

chmod -x /var/www/index.html

chmod -x /var/www/c_sub_modulo.html

cd /var/www/cgi-bin && chmod +x *.cgi

Comandos de compilación del make:

gcc -o a_info_recursos.cgi a_info_recursos.c -lm

gcc -o b_get_telemetry.cgi b_get_telemetry.c

gcc -o b_get_datta.cgi b_get_datta.c

gcc -o b_erase_datta.cgi b_erase_datta.c

gcc -o c_modulos.cgi c_modulos.c -lm

gcc -o c_ins_modulo.cgi c_ins_modulo.c -llamada

gcc -o c_qui_modulo.cgi c_qui_modulo.cgi

Inicio:

Si por alguna razón el sistema no se inicia solo al inicio, puede hacerse manualmente con el comando:

/usr/local/sbin/thttpd -C /etc/thttpd.conf

Y verificar que se encuentra en ejecución con el comando:

ps | grep thttpd

Compilación del módulo: Cross-Compiling

1- En el Sistema Huésped, **instalar** las herramientas de compilación:

```
sudo apt-get install build-essential
```

2- **Descargar** el código fuente del proyecto yocto-galileo desde:

https://software.intel.com/es-es/iot/hardware/galileo/downloads

ARCHIVOS DE CONFORMIDAD Y PAQUETE DE ASISTENCIA PARA PLACA
ARCHIVOS DE CONFORMIDAD DE LICENCIA PÚBLICA GENERAL

ARCHIVOS FUENTE

3- **Descomprimir** los archivos fuentes en el home del sistema huésped en que se llevara a cabo la compilación. No hacer esto en particiones diferentes a la que contiene la raíz del sistema ya que generará errores en los enlaces simbólicos que contiene el proyecto.

4- Dentro de la carpeta que se genera luego de descomprimir, entrar a la carpeta poky/meta-intel-galileo/recipes-kernel/ y **crear un directorio** con el nombre del módulo. Dentro de la carpeta copiar el código fuente del módulo junto al archivo de configuración de extensión .bb que utiliza bitbake para llevar a cabo la compilación. (En el directorio poky/meta-skeleton/recipes-kernel/ se encuentra desarrollado un módulo "hello" junto con el archivo de configuración de bitbake correspondiente que puede ser usado como guía.

5- **Copiar** el directorio del módulo con sus archivos dentro del directorio build_galileo que se encuentra en la carpeta que se genera al descomprimir el proyecto junto a la carpeta poky.

6- Dentro de la carpeta poky **ejecutar**

```
source oe-init-build-env ../build_galileo/
```

Dicho comando nos cambia de posición al directorio build_galileo y crea variables de entorno que serán usadas durante el proceso de compilación.

7- Para comenzar a descargar todos los archivos fuentes y dependencias y generar el build del proyecto completo, y compilar el módulo, **ejecutar**

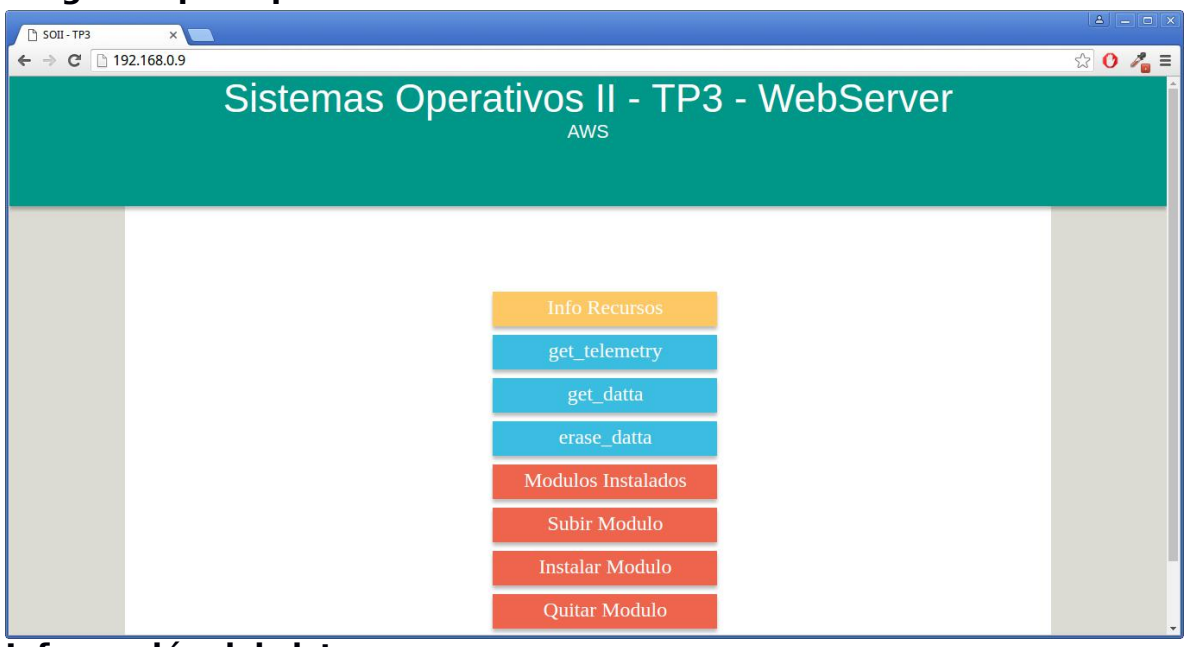
```
bitbake hello
```

Cabe aclarar que el proceso dependiendo de la velocidad de conexión a la red y del hardware del sistema en que se compila puede tomar mas de una o dos horas, y que se requieren como mínimo 10 gb de espacio libre en disco.

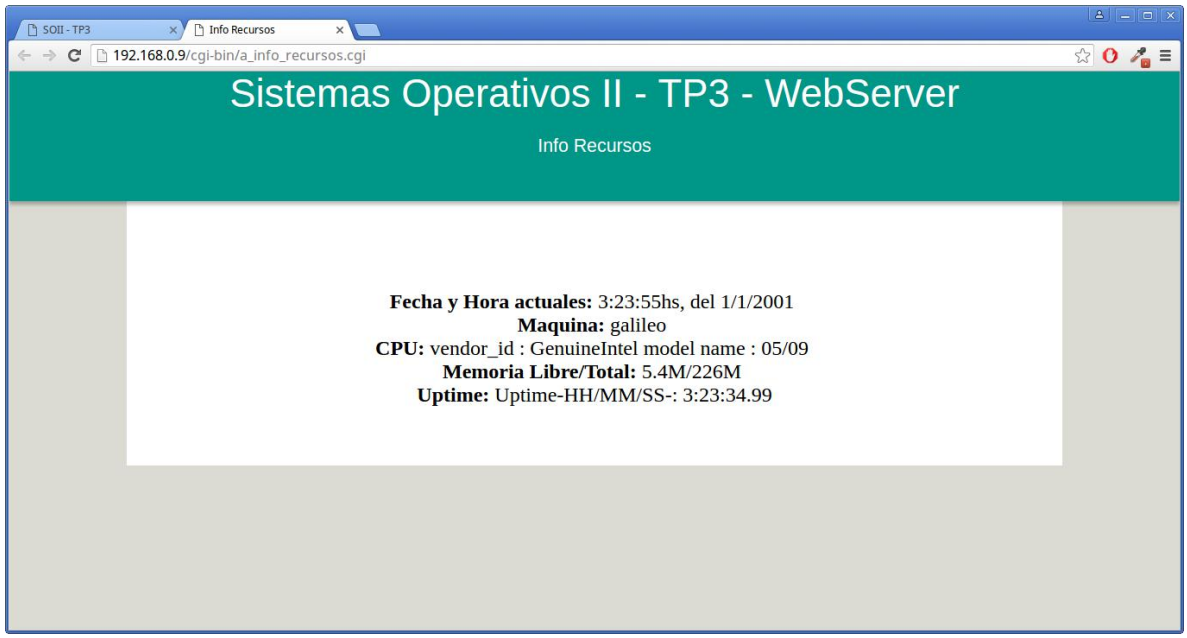
8- Una vez finalizado el proceso, el modulo compilado se encuentra en:

/build_galileo/tmp/work/quark-poky-linux/hello-mod/0.1-r0/

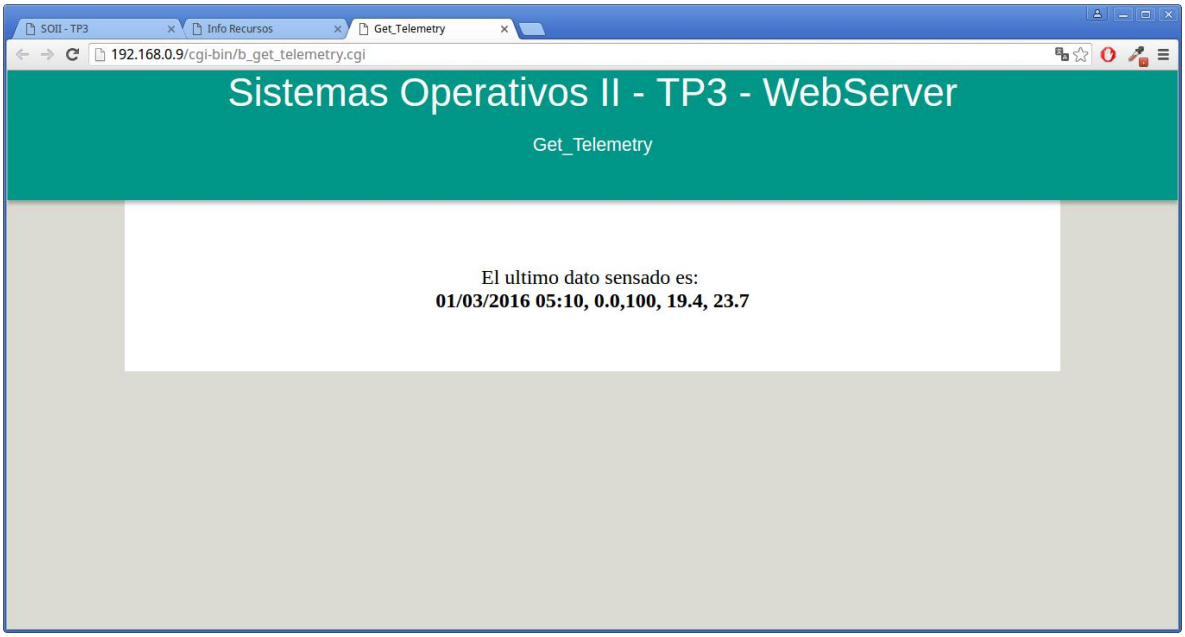
Programa principal:



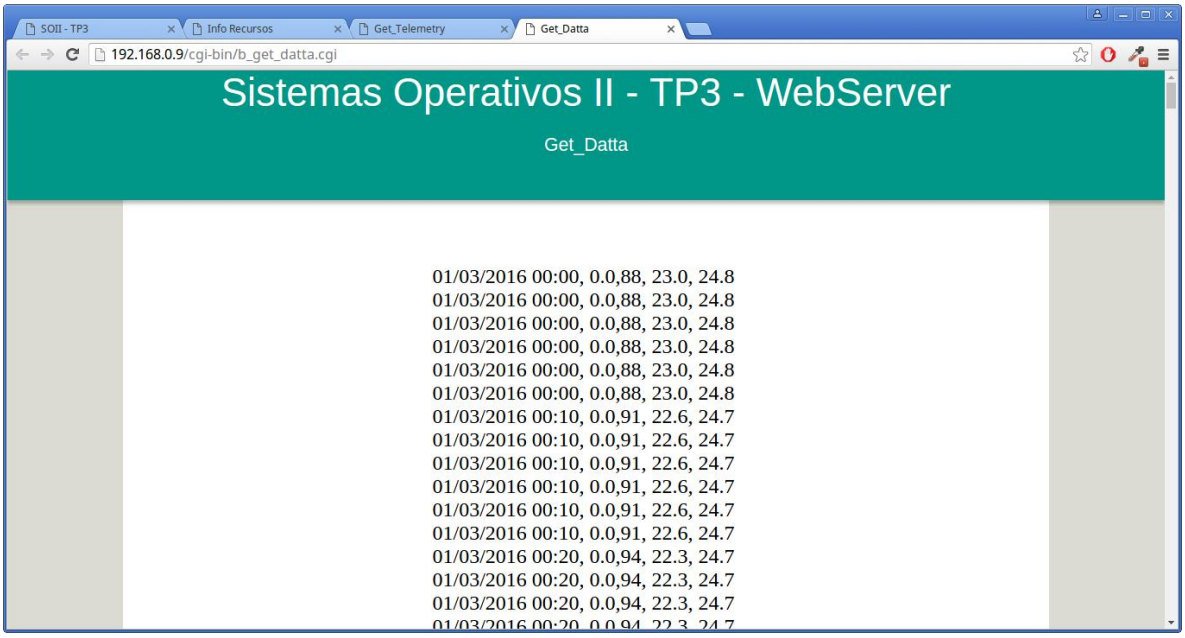
Información del sistema



Get_Telemetry

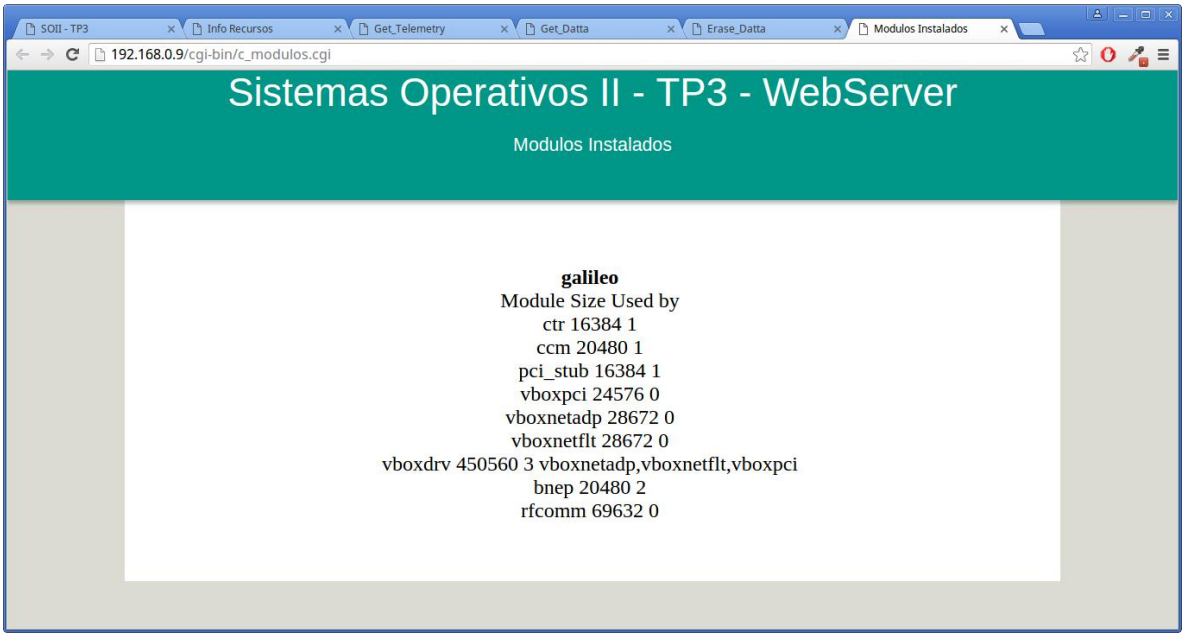


Get_Datta

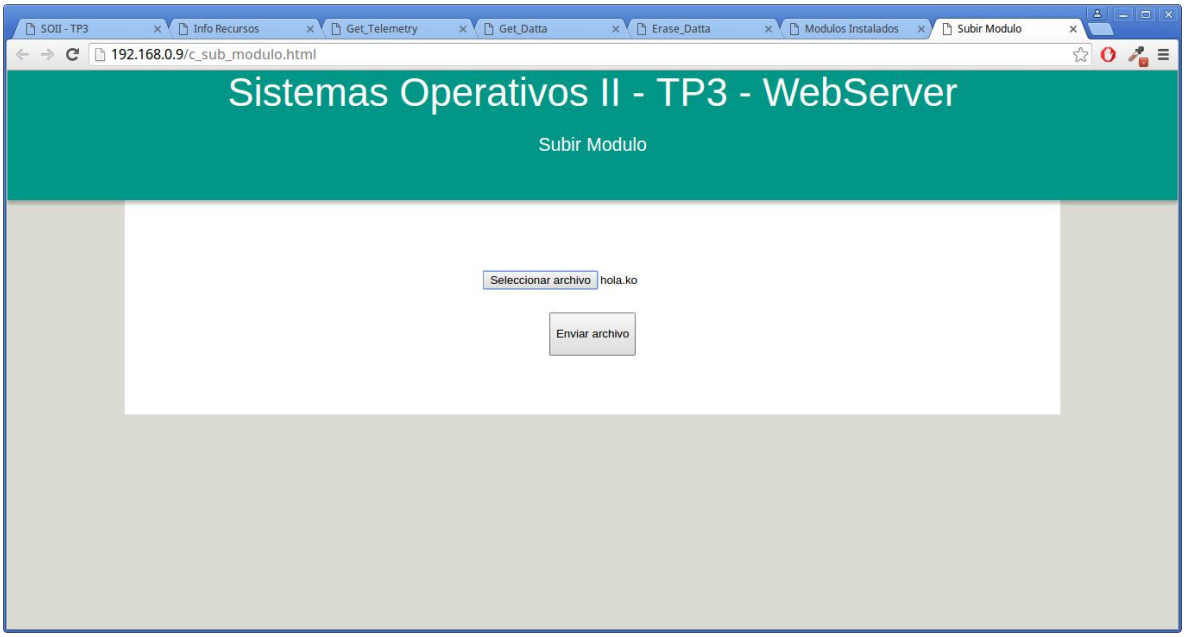


Erase_Datta

Ver Módulos Instalados



Subir Módulo



Todo el sistema montado se encuentra disponible para descarga en la dirección:

https://drive.google.com/folderview?id=0B_XQgaO4OOKOVjVfbmg5STVPUzA&usp=sharing

Conclusión

En este trabajo se aprendió programación en CGI a través de un webserver. Sirvió para descubrir la gran cantidad de servidores disponibles que hay, como Apache, Xampp, AppServ, Monkey, Thttpd, Lighttpd, etc. Se reforzó el procedimiento para compilar y subir un módulo Linux, y se aprendió a trabajar con la placa Intel Galileo I, instalar su Firmware, sistema operativo, etc.
Hubo una gran cantidad de dificultades que superar, sobre todo relacionado con la instalación y puesta a punto del webserver, y el solucionar bastantes errores para esto.