



Universidad Nacional  
de Córdoba

# Cátedra de Sistemas Operativos II

## Trabajo Práctico N° I

**ZIMMEL CECCÓN, Ezequiel José**

[https://gitlab.com/sistiop2/tp1\\_socket](https://gitlab.com/sistiop2/tp1_socket)

2 de febrero de 2020

# Índice

|  |           |
|--|-----------|
| <b>Introducción</b>                    | <b>3</b>  |
| Propósito                              | 3         |
| Ámbito del Sistema                     | 3         |
| Definiciones, Acrónimos y Abreviaturas | 3         |
| Referencias                            | 4         |
| Descripción General del Documento      | 4         |
| <b>Descripción General</b>             | <b>4</b>  |
| Perspectiva del Producto               | 4         |
| Funciones del Producto                 | 4         |
| Características de los Usuarios        | 5         |
| Restricciones                          | 6         |
| Suposiciones y Dependencias            | 6         |
| Requisitos Futuros                     | 6         |
| <b>Requisitos Específicos</b>          | <b>7</b>  |
| Interfaces Externas                    | 7         |
| Funciones                              | 7         |
| Requisitos Funcionales                 | 7         |
| Requisitos de Rendimiento              | 9         |
| Restricciones de Diseño                | 9         |
| Atributos del Sistema                  | 9         |
| <b>Diseño de solución</b>              | <b>10</b> |
| <b>Implementación y Resultados</b>     | <b>12</b> |
| UNIX                                   | 12        |
| INET                                   | 14        |
| <b>Conclusiones</b>                    | <b>16</b> |

# Introducción

Los sockets son una abstracción de comunicación entre procesos (IPC) que, en un sistema tipo UNIX, se implementan en un descriptor de archivo, sobre el cual se envía o recibe información. A su vez, permiten la comunicación entre sistemas remotos, mediante su implementación sobre el stack TCP/IP. Son una herramienta muy importante, sumamente utilizada en la mayoría de las aplicaciones de red, y uno de los pilares de la comunicación entre procesos.

En este documento se encuentra la especificación de los requerimientos de la aplicación Sockets correspondiente al Trabajo Práctico N° 1. Ésto es, se establecen las características que debe tener el sistema o las restricciones que debe satisfacer para la aceptación del cliente.

## Propósito

El propósito de este Trabajo Práctico es aprender a utilizar la estructura de sockets de Linux, sus características y limitaciones, el uso de sus diferentes tipos (orientados o no a la conexión), y su posterior implementación y testing.

Los conceptos deben aplicarse a un caso particular referido a una estación terrestre (servidor), que debe interactuar con un satélite (cliente), y entre los mismos enviar o recibir distintos datos, como la información del satélite, archivos de actualización de firmware o la imagen de la tierra vista desde el satélite.

## Ámbito del Sistema

Bajo implementación de socket INET, el sistema está conformado por dos partes: la aplicación servidor, que se ejecuta desde la computadora del usuario, y la aplicación cliente, que se ejecuta en una placa Raspberry Pi 3. Ambos se comunican por medio de sockets INET.

Por otro lado, bajo implementación de socket UNIX, el sistema se conforma por una única parte, compartiendo un archivo a los fines de la comunicación entre el proceso cliente y el proceso servidor.

## Definiciones, Acrónimos y Abreviaturas

- **TCP:** *Transmission Control Protocol* (Protocolo de Control de Transmission).
- **UDP:** *User Datagram Protocol* (Protocolo de Datagramas de Usuarios).
- Sockets: "canal de comunicación" entre dos programas que corren sobre ordenadores distintos o incluso en el mismo ordenador.
- **INET:** socket de internet.

- UNIX: socket interprocesos.

## Referencias

1. Michael Kerrisk, The Linux Programming Interface, 2010, Adison-Wesley
2. Imagen del satélite Goes16,  
<https://cdn.star.nesdis.noaa.gov/GOES16/ABI/FD/13/20190811400GOES16-ABI-FD-13-10848x10848.jpg.zip>
3. Making The Best Use of C,  
[https://www.gnu.org/prep/standards/html\\_node/Writing-C.html](https://www.gnu.org/prep/standards/html_node/Writing-C.html)
4. L.Torvalds, Et al.,  
<https://github.com/torvalds/linux/tree/master/Documentation/process>

## Descripción General del Documento

El propósito de este documento es proporcionar al lector un entendimiento básico de los requisitos y alcance del proyecto, así como los pasos que se siguieron para su diseño e implementación.

El documento está organizado en 6 secciones principales:

- Descripción de las generalidades del documento y del proyecto.
- Descripción de la perspectiva del producto, sus funciones y características del usuario y sus restricciones.
- Referencias a los requerimientos del producto, interfaces y casos de uso, incluyendo diagramas y gráficos que facilitan su entendimiento.
- Diseño de la solución.
- Implementación de la solución y resultados obtenidos.
- Conclusiones a las tareas realizadas.

## Descripción General

En la presente sección se explicarán los distintos aspectos del trabajo, como ser las funciones que ofrece el producto, la perspectiva del mismo, las características de los usuarios y los requisitos para futuras iteraciones.

### Perspectiva del Producto

El producto está orientado a la comunicación de satélites con la estación terrestre para que ésta pueda recibir las distintas imágenes satelitales, el estado del satélite y, en caso de ser necesario, actualizar el firmware del satélite desde la tierra.

### Funciones del Producto

El producto provee las siguientes funciones:

- Conectividad usuario-servidor mediante IP y puerto (socket INET), o mediante archivo (socket UNIX).
- Autenticación de usuario para brindar seguridad en los datos. Característica del servidor, que verificará la existencia del usuario y contraseña, dando acceso a la terminal interactiva.
- Control de acceso con 3 intentos de autenticación.
- Terminal interactiva que posibilita la comunicación del servidor con los satélites.
- Actualización remota del firmware del satélite.
- Envío de imágenes satelitales a la estación.
- Obtención de los datos del satélite (telemetría).

## Características de los Usuarios

Los Usuarios deberán contar con conocimiento básico para manejarse con la consola, ya que no se cuenta con una interfaz gráfica.

| FUNCIÓN  | CLIENTE               | SERVIDOR                  |
|--|-----------------------|---------------------------|
| <b>Autenticación</b><br>Solicitud de usuario y contraseña<br>Validación de credenciales<br>Notificación en control de acceso<br>Inicialización de socket STREAM  |                       | ✓<br>✓<br>✓<br>✓          |
| <b>Establecer conexión satélite-estación</b><br>Solicitud de conexión con estación terrestre<br>Validación de parámetros de conexión<br>Inicialización de socket STREAM  | ✓<br>✓<br>✓           |                           |
| <b>Terminal</b><br>Interactiva<br>Informativa  | ✓                     | ✓<br>✓                    |
| <b>Actualización de Firmware</b><br>Ingreso de comando <i>firmware_update</i><br>Validación del comando<br>Obtención del tamaño del nuevo Firmware<br>Envío del archivo de actualización<br>Recepción del archivo y ajustes de ejecución<br>Ejecución del binario recibido | ✓<br>✓<br>✓<br>✓<br>✓ | ✓<br>✓<br>✓<br>✓          |
| <b>Obtener datos telemétricos</b><br>Ingreso de comando <i>obtener_telemetria</i><br>Validación del comando<br>Inicialización de socket DATAGRAM<br>Obtención de los datos<br>Envío de datos<br>Recepción de datos<br>Cierre de socket DATAGRAM                            | ✓<br>✓<br>✓<br>✓<br>✓ | ✓<br>✓<br>✓<br><br>✓<br>✓ |

|   |                  |                       |
|---|------------------|-----------------------|
| <b>Iniciar escaneo de superficie terrestre</b><br>Ingreso de comando <i>start_scanning</i><br>Validación del comando<br>Obtención del tamaño del archivo<br>Envío del archivo<br>Notificación de progreso en la transferencia<br>Recepción y almacenamiento del archivo | ✓<br>✓<br>✓<br>✓ | ✓<br>✓<br>✓<br>✓<br>✓ |
| <b>Listado de opciones</b><br>Ingreso de comando <i>opciones</i><br>Validación del comando<br>Despliegue de opciones soportadas   |                  | ✓<br>✓<br>✓           |
| <b>Finalización de conexión con satélite</b><br>Ingreso de comando <i>sat_logoff</i><br>Validación del comando<br>Cierre de socket STREAM   | ✓<br>✓           | ✓<br>✓<br>✓           |

## Restricciones

Las aplicaciones deben compilarse y ejecutarse en sistemas operativos UNIX con permisos para levantar sockets y nuevos procesos.

Se requiere de una terminal con requerimientos de hardware capaz de correr el programa del servidor y, de la misma forma, una terminal con requerimientos de hardware, con MMU, que ejecute el programa del cliente.

## Suposiciones y Dependencias

- El proyecto se ejecutará sobre UNIX.
- Existe conectividad entre el servidor y el cliente, para poder realizar la comunicación.
- El proyecto debe funcionar en una distribución Linux estándar, utilizando *make* para compilar el código fuente.
- Tanto el servidor como el cliente tendrán acceso a internet y una dirección Ipv4.
- El servidor dispondrá de memoria suficiente para almacenar las distintas imágenes satelitales.

## Requisitos Futuros

Se deberán cumplir los siguientes requisitos para futuras iteraciones:

- Soporte para múltiples conexiones satelitales.
- Cifrado de los datos telemétricos.
- Auto diagnóstico del estado del satélite.
- Modo *Stand By* del satélite.

# Requisitos Específicos

## Interfaces Externas

- Dispositivo del cliente; debe tener instalado un sistema operativo con kernel linux.
- Dispositivo del servidor; debe tener instalado un sistema operativo con kernel linux.
- Interfaz de usuario, la cual se ofrece al usuario a modo de Prompt en una consola. Simplifica el uso del programa por parte del usuario, reduciendo el volumen de errores.
- Sockets empleados para la comunicación entre procesos.

## Funciones

Una vez que el usuario se ha autenticado en el servidor, el mismo acepta los siguientes comandos:

- **update\_firmware**: envía un archivo binario al satélite con una actualización de software (firmware). Una vez actualizado, se debe reiniciar el satélite con la nueva actualización.
- **obtener\_telemetría**: el satélite le envía a la estación terrena la siguiente información:
  - ID: Nombre del satélite.
  - Uptime: hora y fecha de la última actualización.
  - Versión: versión actual de firmware.
  - CPU: consumo de memoria y CPU.
- **start\_scanning**: inicia un escaneo de toda la cara de la Tierra. Cada escaneo se envía a la estación terrestre.
- **opciones**: muestra un mensaje de ayuda con los comandos disponibles.
- **sat\_logoff**: cierra la sesión actual con el satélite.

## Requisitos Funcionales

|          |   |
|----------|---|
| Conexión | <ol style="list-style-type: none"><li>1. La conexión debe ser libre de errores entre el cliente y el servidor para la transferencia de información.</li><li>2. Debe notificarse el estado de conexión, direcciones de los extremos del enlace PtP.</li><li>3. La inicialización del socket STREAM debe ser posterior a la validación de las credenciales de acceso al servidor.</li><li>4. El cliente debe detectar una denegación de conexión al servidor.</li></ol> |
|----------|---|

|                                   |  |
|-----------------------------------|--|
| <b>Prompt interactivo</b>         | <ol style="list-style-type: none"> <li>1. No es posible ingresar comandos si no hay clientes conectados.</li> <li>2. El servidor da acceso al prompt solo si se validan las credenciales de acceso.</li> <li>3. Los comandos se ingresan sólo si termino la ejecución del comando anterior.</li> <li>4. Deben ignorarse aquellos comandos desconocidos.</li> <li>5. Debe salir del prompt si pierde conectividad con el cliente.</li> </ol>  |
| <b>Autenticación</b>              | <ol style="list-style-type: none"> <li>1. El servidor debe poder verificar al usuario que intenta acceder al servidor en una base de datos.</li> <li>2. Debe notificarse el estado de autenticación, así como los intentos restantes en el acceso de credenciales.</li> <li>3. Debe de ocultarse la información durante el ingreso de la contraseña.</li> <li>4. En caso de fallo de autenticación 3 veces seguidas el servidor debe terminar la aplicación</li> <li>5. La base de datos debe estar guardada en un archivo de texto plano.</li> <li>6. No se aceptan ni envían comandos previo a autenticar al usuario.</li> </ol> |
| <b>Actualización del Firmware</b> | <ol style="list-style-type: none"> <li>1. Solo debe ser aceptado el comando <i>update_firmware</i> para activar en ambos extremos de la conexión PtP el procedimiento de actualización.</li> <li>2. El archivo a enviar es el firmware por TCP.</li> <li>3. El archivo binario puede reemplazarse respetando formato.</li> <li>4. El cliente debe recibir el archivo, almacenarlo y reiniciarse lanzando su nueva actualización a través del archivo recibido.</li> </ol>  |
| <b>Escaneo</b>                    | <ol style="list-style-type: none"> <li>1. El único comando aceptado para la ejecución de esta función debe <i>start_scanning</i>.</li> <li>2. En el escaneo se debe enviar un archivo de la forma <i>nombre.jpg</i>.</li> <li>3. El archivo debe transmite por TCP.</li> <li>4. El la transmisión de la imagen, el paquete en capa 3 del stack TCP/IP no debe fragmentarse.</li> <li>5. El archivo jpg puede reemplazarse respetando formato.</li> </ol>   |



|                           |  |
|---------------------------|--|
| <b>Obtener telemetría</b> | <ol style="list-style-type: none"> <li>1. El único comando aceptado para esta función debe <i>obtener telemetría</i>.</li> <li>2. Los datos deben enviarse mediante un socket UDP.</li> <li>3. El servidor debe levantar el socket UDP para escuchar al cliente.</li> <li>4. El comando debe enviar información de: ID satélite, uptime, versión del firmware, uso RAM</li> </ol>                                  |
| <b>Cierre de conexión</b> | <ol style="list-style-type: none"> <li>1. El único comando aceptado para esta función es <i>sat_logoff</i>.</li> <li>2. El servidor debe cerrar su socket y el programa debe finalizar.</li> <li>3. El cliente debe cerrar su socket y el programa debe finalizar.</li> <li>4. El usuario debe recibir un mensaje a través del prompt que finalizó el programa y ya no puede interactuar con el prompt.</li> </ol> |

## Requisitos de Rendimiento

- El proceso de enviar la imagen del satélite a la estación terrestre debe ser lo más óptima y en el menor tiempo posible.
- Uso de memoria mínimo y apropiado para que el programa pueda ser corrido en una placa de desarrollo con MMU.

## Restricciones de Diseño

- En el servidor no se podrá conectar más de un usuario por vez.
- Tanto el código del cliente como el del servidor serán desarrollados en lenguaje C.
- Se utilizará el puerto 6020 como puerto para la conexión TCP, puerto en el cual se encontrará, en todo momento, escuchando el servidor una vez realizada la autenticación.
- Se utilizará el puerto 6020 como puerto para la conexión UDP en la transferencia de datos telemétricos.
- El servidor estará montado en una placa de desarrollo que soporte sistemas operativos del tipo GNU/Linux.
- Sintaxis de código del tipo GNU o Linux Kernel.

## Atributos del Sistema

El servidor debe correr en una computadora con distribución Linux, mientras que el cliente (satélite) debe correr sobre una terminal con MMU. A su vez, se destaca la

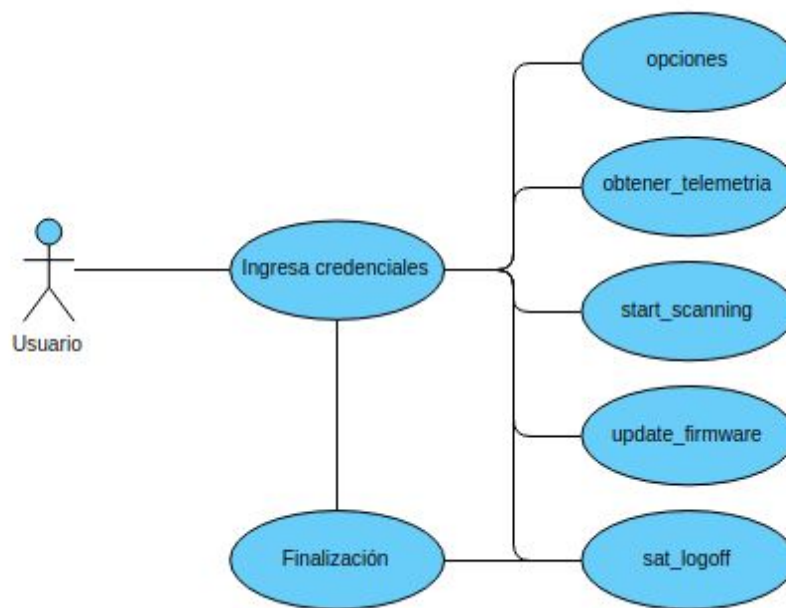
portabilidad del sistema y la capacidad de su utilización en sistemas operativos tipo UNIX en equipos de escasos recursos, ideal para sistemas embebidos.

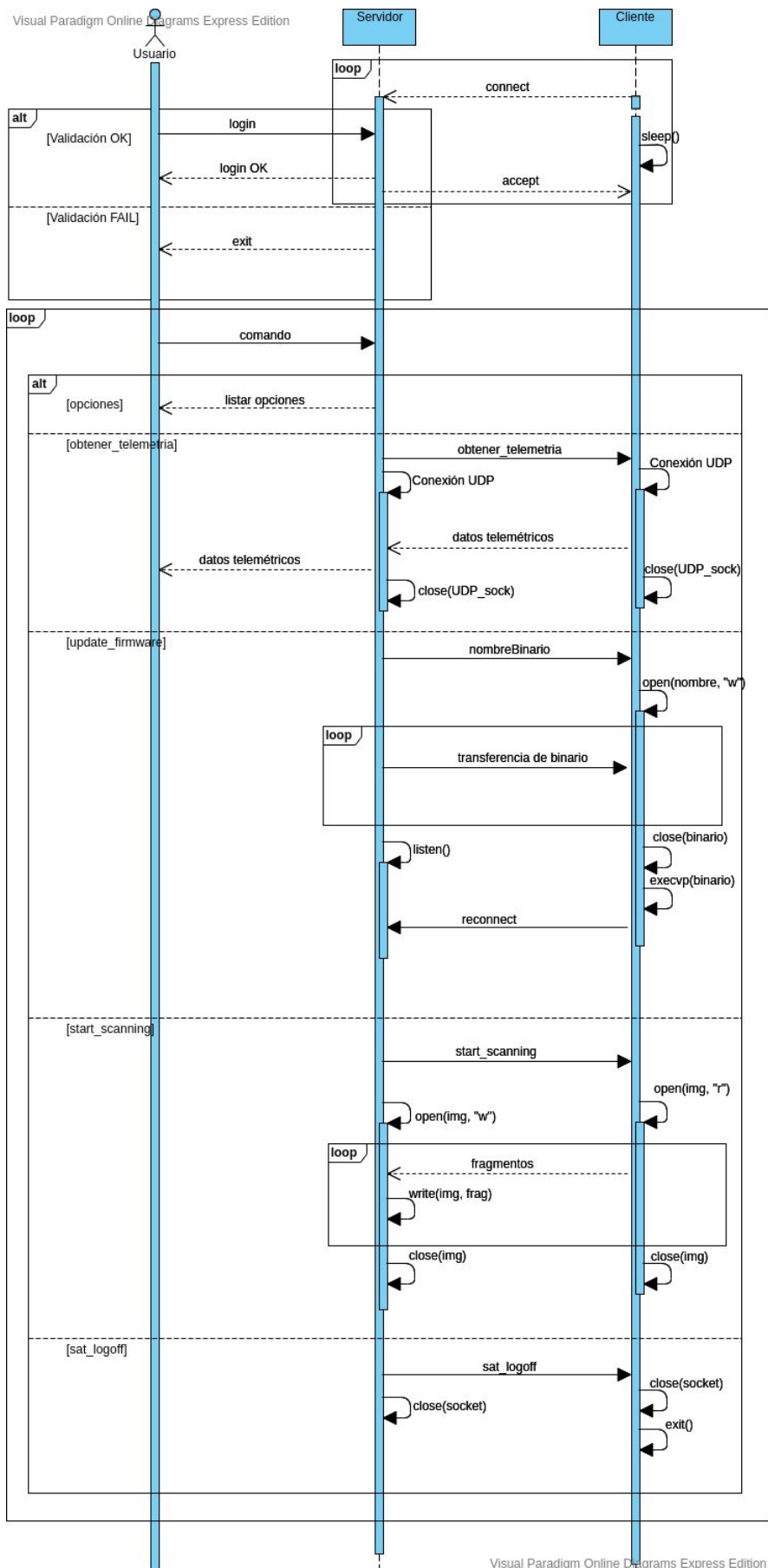
## Diseño de solución

Se desarrollaron dos aplicaciones, una que es ejecutada en el cliente y otra en el servidor.

- *Cliente*: se conecta al servidor y queda a la espera de los comandos, los cuales son introducidos desde el servidor para realizar las distintas funciones.
- *Servidor*: crea los sockets para la conexión y autentifica los usuarios, que luego podrán pedir un dato específico al cliente mediante el ingreso de comandos disponibles.

El primer paso es el logueo del usuario al sistema. Una vez que se validan sus credenciales, el sistema despliega el prompt en donde se puede acceder a todas las funciones provistas.





## Implementación y Resultados

Para la implementación y desarrollo se utilizó el software *Visual Studio Code* y los códigos sockets aportados por la cátedra. En forma complementaria, se crearon funciones, las cuales se utilizaron para el proyecto.

Primero, se procedió a trabajar en la implementación sobre sockets UNIX. Una vez que el trabajo estuvo implementado en dichos sockets, se comprobó la funcionalidad de los mismos. Luego, se procedió a implementar el desarrollo sobre sockets INET.

A continuación, se muestra el funcionamiento de las implementaciones UNIX e INET. Las imágenes ubicadas a la izquierda corresponden al comportamiento del cliente, mientras que, las ubicadas a la derecha, al servidor.

### UNIX

Los programas se ejecutan de la siguiente manera:

- ./cliente <socket UNIX>
  - ./servidor
1. El cliente intentará conectarse al servidor de manera indefinida hasta que este último esté disponible.



```
=====
Cliente inicializado - Intento[1]
Conexion [x]
=====

=====
Cliente inicializado - Intento[2]
Conexion [x]
=====

=====
Cliente inicializado - Intento[3]
Conexion [x]
=====

Inicio del programa Servidor
=====
desconectado~$
```

2. El usuario del programa servidor debe presentar credenciales válidas; en caso de no corresponderse con las almacenadas en la base de datos, se le denegará el acceso una vez excedida la cantidad de reintentos.



```
=====
Cliente inicializado - Intento[1]
Conexion [x]
=====

=====
Cliente inicializado - Intento[2]
Conexion [x]
=====

Inicio del programa Servidor
=====
desconectado~$ login admin@server
[3]Ingrese contraseña: [x]
```

- Una vez que el usuario presenta credenciales válidas, se despliega el estado de la conexión y un prompt para acceder a las opciones del sistema.

```

=====
Cliente inicializado - Intento[1]
Conexion [X]
=====

Cliente inicializado - Intento[2]
Conexion [X]
=====

Cliente inicializado [ID: 25696]
Version Firmware: 1
Conexion [V]
=====
Satelite Activo...
[ ]

Inicio del programa Servidor
=====
desconectado-$ login admin@server
Ingrese contraseña: [V]
Bienvenido admin
Esperando por conexión entrante
Proceso: 25543 - socket disponible: server

SERVIDOR: Nuevo cliente (PID: 25696) conectado

Escriba 'opciones' para listar los comandos disponibles.
admin@server # [ ]

```

- El comando *opciones* lista los comandos disponibles para la interacción con el satélite.

```

=====
Cliente inicializado - Intento[1]
Conexion [X]
=====

Cliente inicializado - Intento[2]
Conexion [X]
=====

Cliente inicializado [ID: 25696]
Version Firmware: 1
Conexion [V]
=====
Satelite Activo...
[ ]

Inicio del programa Servidor
=====
desconectado-$ login admin@server
Ingrese contraseña: [V]
Bienvenido admin
Esperando por conexión entrante
Proceso: 25543 - socket disponible: server

SERVIDOR: Nuevo cliente (PID: 25696) conectado

Escriba 'opciones' para listar los comandos disponibles.
admin@server # opciones

OPCIONES
1)update_firmware
2)start_scanning
3)obtener telemetria
4)opciones
5)sat_logoff

admin@server # [ ]

```

- El comando *obtener\_telemetria* envía los datos del satélite hacia la estación terrestre.

```

=====
Cliente inicializado [ID: 25696]
Version Firmware: 1
Conexion [V]
=====
Satelite Activo...
[ ]

ENVIANDO TELEMETRIA
[1-7] ID satellite: 25696
[2-7] Version Firmware: 1
[3-7] Uptime : 0 dias, 11:58:45
[4-7] Boot Time: Sun Feb 2 07:07:28 2020
[5-7] Hostname: ezequiel-Note
[6-7] MemTotal: 15882 - MemFree: 5594
[7-7] CPU: 12.9292%
=====
Satelite Activo...
[ ]

4)opciones
5)sat_logoff

admin@server # obtener_telemetria
Enviando orden OBTENER TELEMETRIA
Usando socket: server_UDP
=====

OBTENER TELEMETRIA
[1-7] ID satellite: 25696
[2-7] Version Firmware: 1
[3-7] Uptime : 0 dias, 11:58:45
[4-7] Boot Time: Sun Feb 2 07:07:28 2020
[5-7] Hostname: ezequiel-Note
[6-7] MemTotal: 15882 - MemFree: 5594
[7-7] CPU: 12.9292%
=====

admin@server # [ ]

```

- Ejecutando el comando *start\_scanning*, se transfiere la imagen satelital a la estación terrestre. Se muestra información del tamaño en bytes de la imagen, la cantidad de paquetes a enviar y el porcentaje de avance durante la transmisión.

```

=====
Satelite Activo...
=====

START SCANNING

Tamaño de Imagen: 77959899
N° de paquetes a enviar : 51973
Finalizado envio de Imagen
=====
Satelite Activo...
[

admin@server # start_scanning
Enviando orden START SCANNING
=====

START SCANNING

N° de paquetes a recibir: 51973
[51973 - 51973] [100%] Finalizada la recepcion de Imagen
=====
admin@server # [

```

7. La actualización del firmware del satélite se realiza mediante el comando *update\_firmware*. Una vez que el satélite recibe el nuevo binario, procede a reiniciarse, cargando el nuevo firmware y re-enlazándose a la estación terrestre.

```

=====
Satelite Activo...
=====

UPDATE FIRMWARE

N° de paquetes a recibir: 283
Reiniciando...
=====

Cliente inicializado [ID: 25696]
Version Firmware: 2
Conexion [v]
=====
Satelite Activo...
[

admin@server # update_firmware
Enviando orden UPDATE FIRMWARE
=====

UPDATE FIRMWARE

Tamaño del binario: 22696
N° de paquetes a enviar: 283
=====

Cerrando comunicacion con cliente.
Esperando por conexión entrante

SERVIDOR: Nuevo cliente (PID: 25696) conectado

Escriba 'opciones' para listar los comandos disponibles.
admin@server # [

```

```

Conexion [v]
=====
Satelite Activo...
=====

ENVIANDO TELEMETRIA

[1-7] ID satellite: 25696
[2-7] Version Firmware: 2
[3-7] Uptime : 0 dias, 12:03:55
[4-7] Boot Time: Sun Feb 2 07:07:28 2020
[5-7] Hostname: ezequiel-Note
[6-7] MemTotal: 15882 - MemFree: 5365
[7-7] CPU: 12.946%
=====
Satelite Activo...
[

admin@server # obtener telemetria
Enviando orden OBTENER TELEMETRIA
Usando socket: server_UDP
=====

OBTENER TELEMETRIA

[1-7] ID satellite: 25696
[2-7] Version Firmware: 2
[3-7] Uptime : 0 dias, 12:03:55
[4-7] Boot Time: Sun Feb 2 07:07:28 2020
[5-7] Hostname: ezequiel-Note
[6-7] MemTotal: 15882 - MemFree: 5365
[7-7] CPU: 12.946%
=====
admin@server # [

```

8. La desconexión de la estación con el satélite se realiza mediante *sat\_logoff*. Una vez cerrada en ambos extremos la conexión, la estación queda a la espera de nuevas conexiones entrantes.

```

=====
Satelite Activo...

Cerrando comunicacion.

admin@server # sat_logoff
Cerrando comunicacion con cliente.
Esperando por conexión entrante

```

## INET

Los programas se ejecutan de la siguiente manera:

- ./cliente <IP servidor>:<Puerto servidor>
- ./servidor

Las características son idénticas a las desarrolladas en socket UNIX.

1. El proceso de validación de credenciales y conexión con el servidor son idénticas que en socket UNIX. La diferencia radica el parámetro pasado para la conexión, siendo, en este caso, la dirección IP y puesto del programa servidor.



```

=====
Cliente inicializado [ID: 1290] [192.168.0.14]
Version Firmware: 1
Conexion [V]
=====
Satelite Activo...
[ ]

=====
Inicio del programa Servidor
=====
desconectado-$ login admin
Ingreso contraseña: [V]
Bienvenido admin
Esperando por conexion entrante
Proceso: 28550 - socket disponible: 192.168.1.4:6020

SERVIDOR: Nuevo cliente (PID: 1290) conectado desde 192.168.1.5:34900

Escriba 'opciones' para listar los comandos disponibles.
admin@192.168.1.4:6020 # [ ]

```

## 2. Ejecución del comando *obtener\_telemetria*.

```

ENVIANDO TELEMETRIA

Puerto a usar: 6020
[1-7] ID satellite: 1290
[2-7] Version Firmware: 1
[3-7] Uptime : 0 dias, 0:29:47
[4-7] Boot Time: Tue Jan 28 16:59:54 2020
[5-7] Hostname: raspberrypi
[6-7] MemTotal: 926 - MemFree: 385
[7-7] CPU: 2.32409%

=====
Satelite Activo...
[ ]

Visor de documentos

=====
SERVIDOR: Nuevo cliente (PID: 1290) conectado desde 192.168.1.5:34900

Escriba 'opciones' para listar los comandos disponibles.
admin@192.168.1.4:6020 # obtener_telemetria
Enviando orden OBTENER TELEMETRIA
Usando socket: 192.168.1.4:6020

=====
OBTENER TELEMETRIA

[1-7] ID satellite: 1290
[2-7] Version Firmware: 1
[3-7] Uptime : 0 dias, 0:29:47
[4-7] Boot Time: Tue Jan 28 16:59:54 2020
[5-7] Hostname: raspberrypi
[6-7] MemTotal: 926 - MemFree: 385
[7-7] CPU: 2.32409%

=====
admin@192.168.1.4:6020 # [ ]

```

## 3. Ejecución del comando *start\_scanning*. Se observa que los paquetes transmitidos no son fragmentados, cumpliendo con uno de los requerimientos.

```

=====
Satelite Activo...
=====

START SCANNING

Tamaño de Imagen: 77959899
N° de paquetes a enviar : 51973
Finalizado envio de Imagen

=====
Satelite Activo...
[ ]

=====
admin@192.168.1.4:6020 # start_scanning
Enviando orden START SCANNING

=====
START SCANNING

N° de paquetes a recibir: 51973
[51973 - 51973] [100%] Finalizada la recepcion de Imagen
=====
admin@192.168.1.4:6020 # [ ]

```

| No. | Time        | Source      | Destination | Protocol | Length | Info   |
|-----|-------------|-------------|-------------|----------|--------|--|
| 128 | 0.069787328 | 192.168.1.5 | 192.168.1.4 | TCP      | 1290   | 34900 → 6020 [ACK] Seq=115373 Ack=15 Win=501 Len=1224 T... |
| 129 | 0.069845790 | 192.168.1.4 | 192.168.1.5 | TCP      | 66     | 6020 → 34900 [ACK] Seq=15 Ack=116597 Win=135 Len=0 TSva... |
| 130 | 0.069910718 | 192.168.1.5 | 192.168.1.4 | TCP      | 1514   | 34900 → 6020 [ACK] Seq=116597 Ack=15 Win=501 Len=1448 T... |
| 131 | 0.070033341 | 192.168.1.5 | 192.168.1.4 | TCP      | 1514   | 34900 → 6020 [ACK] Seq=118045 Ack=15 Win=501 Len=1448 T... |
| 132 | 0.070050947 | 192.168.1.4 | 192.168.1.5 | TCP      | 66     | 6020 → 34900 [ACK] Seq=15 Ack=119493 Win=120 Len=0 TSva... |
| 133 | 0.070157163 | 192.168.1.5 | 192.168.1.4 | TCP      | 1514   | 34900 → 6020 [ACK] Seq=119493 Ack=15 Win=501 Len=1448 T... |
| 134 | 0.070285466 | 192.168.1.5 | 192.168.1.4 | TCP      | 1514   | 34900 → 6020 [ACK] Seq=120941 Ack=15 Win=501 Len=1448 T... |
| 135 | 0.070403090 | 192.168.1.5 | 192.168.1.4 | TCP      | 1514   | 34900 → 6020 [ACK] Seq=122389 Ack=15 Win=501 Len=1448 T... |
| 136 | 0.070527541 | 192.168.1.5 | 192.168.1.4 | TCP      | 1514   | 34900 → 6020 [ACK] Seq=123837 Ack=15 Win=501 Len=1448 T... |
| 137 | 0.070562676 | 192.168.1.4 | 192.168.1.5 | TCP      | 66     | 6020 → 34900 [ACK] Seq=15 Ack=125285 Win=89 Len=0 TSva...  |
| 138 | 0.070649734 | 192.168.1.5 | 192.168.1.4 | TCP      | 1514   | 34900 → 6020 [ACK] Seq=125285 Ack=15 Win=501 Len=1448 T... |
| 139 | 0.070774755 | 192.168.1.5 | 192.168.1.4 | TCP      | 1514   | 34900 → 6020 [ACK] Seq=126733 Ack=15 Win=501 Len=1448 T... |

▶ Frame 135: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0  
 ▶ Ethernet II, Src: Raspberr\_e2:0f:71 (b8:27:eb:e2:0f:71), Dst: QuantaCo\_6d:04:18 (a8:1e:84:6d:04:18)  
 ▶ Internet Protocol Version 4, Src: 192.168.1.5, Dst: 192.168.1.4  
   0100 .... = Version: 4  
   .... 0101 = Header Length: 20 bytes (5)  
   ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
   Total Length: 1500  
   Identification: 0x8657 (34391)  
   ▶ Flags: 0x4000, Don't fragment  
   ...0 0000 0000 0000 = Fragment offset: 0  
   Time to live: 64  
   Protocol: TCP (6)  
   Header checksum: 0x2b6b [validation disabled]  
   [Header checksum status: Unverified]

## 4. Ejecución del comando *update\_firmware*.

```

=====
Satelite Activo...
=====
UPDATE FIRMWARE
N° de paquetes a recibir: 239
Reiniciando...
=====
=====
  Cliente inicializado [ID: 1290] [192.168.0.14]
  Version Firmware: 2
  Conexion [.]
=====
Satelite Activo...
[.]
=====
admin@192.168.1.4:6020 # update_firmware
Enviando orden UPDATE FIRMWARE
=====
UPDATE FIRMWARE
Tamaño del binario: 19192
N° de paquetes a enviar: 239
=====
Cerrando comunicacion con cliente.
Esperando por conexión entrante
SERVIDOR: Nuevo cliente (PID: 1290) conectado desde 192.168.1.5:34902
Escriba 'opciones' para listar los comandos disponibles.
admin@192.168.1.4:6020 # [.]

```

5. Ejecución del comando *obtener\_telemetria*, posterior a la actualización del firmware.

```

=====
Satelite Activo...
=====
ENVIANDO TELEMETRIA
Puerto a usar: 6020
[1-7] ID satellite: 1290
[2-7] Version Firmware: 2
[3-7] Uptime : 0 dias, 0:43:47
[4-7] Boot Time: Tue Jan 28 16:59:54 2020
[5-7] Hostname: raspberrypi
[6-7] MemTotal: 926 - MemFree: 385
[7-7] CPU: 2.09652%
=====
Satelite Activo...
[.]
=====
Escriba 'opciones' para listar los comandos disponibles.
admin@192.168.1.4:6020 # obtener_telemetria
Enviando orden OBTENER TELEMETRIA
Usando socket: 192.168.1.4:6020
=====
OBTENER TELEMETRIA
[1-7] ID satellite: 1290
[2-7] Version Firmware: 2
[3-7] Uptime : 0 dias, 0:43:47
[4-7] Boot Time: Tue Jan 28 16:59:54 2020
[5-7] Hostname: raspberrypi
[6-7] MemTotal: 926 - MemFree: 385
[7-7] CPU: 2.09652%
=====
admin@192.168.1.4:6020 # [.]

```

6. Ejecución del comando *sat\_logoff*.

```

=====
Satelite Activo...
=====
Cerrando comunicacion.
pi@raspberrypi:~/Desktop/Internet/Cliente1 $ [.]
=====
admin@192.168.1.4:6020 # sat_logoff
Cerrando comunicacion con cliente.
Esperando por conexión entrante
[.]

```

## Conclusiones

Por medio de la resolución del trabajo práctico, se logró llevar a cabo una conexión de dos terminales distintas por medio de sockets, logrando así realizar un proyecto de la vida real. Se puso en práctica una forma de IPC, tanto para comunicación de procesos por Internet como a nivel local.

Además, se profundizaron los conocimientos de programación en C, uso de punteros, acceso a archivos, entre otros.