Proyecto 3
Desbloqueador de teléfono

1st Acevedo Rodríguez Kevin
Instituto Tecnológico de Costa Rica
Principios de Sistemas Operativos
Cartago, Costa Rica
kevinar51@estudiantec.cr

2nd Camacho Hernández José Julián
Instituto Tecnológico de Costa Rica
Principios de Sistemas Operativos
Cartago, Costa Rica
jcamacho341@estudiantec.cr

3rd Venegas Vega Jose Agustín
Instituto Tecnológico de Costa Rica
Principios de Sistemas Operativos
Cartago, Costa Rica
joseagustinveneveg@estudiantec.cr

4th Solís Arguello Juan Antonio
Instituto Tecnológico de Costa Rica
Principios de Sistemas Operativos
Cartago, Costa Rica
JuanSa@estudiantec.cr

***Resumen*—This document presents the development of a project aimed at implementing a robotic finger prototype. The project involves the creation of a device driver and the necessary software layers to enable seamless interaction between the hardware and the operating system. The project focuses on strengthening hardware and software integration techniques through the device driver, while also detailing the communication process required for proper interaction between the operating system and the computer's hardware.**

***Index Terms*—Driver, Hardware,Sistema Operativo, Controlador, Arduino.**

I. Introducción

En el presente documento, se abordará el desarrollo de un proyecto que tiene como objetivo la implementación de un prototipo de dedo robótico. Este prototipo requerirá de un controlador de dispositivo (device driver) y diversas capas de software que permitirán la interacción entre el hardware y el sistema operativo. Se enfocará en el fortalecimiento de técnicas de integración entre hardware y software a través del controlador de dispositivo, además de detallar el proceso de comunicación necesario entre el sistema operativo y el hardware del ordenador para lograr una interacción adecuada.

El proyecto se centra en cuatro áreas fundamentales de los sistemas operativos: los controladores de dispositivo, el hardware, la sincronización y el procesamiento distribuido. Para su correcta ejecución, se ha diseñado una estructura compuesta por distintas capas, las cuales se muestran en la figura 1. Estas capas son de suma importancia, ya que requieren insumos específicos y proporcionan los elementos necesarios para llevar a cabo un procesamiento óptimo.

II. Ambiente de desarrollo

A continuación se detalla la configuración básica necesaria para la ejecución del proyecto. El mismo fue desarrollado en un ambiente Linux, por tanto fue posible la utilización del conjunto de herramientas base que ofrece dicha plataforma como GNU, *gcc*, entre otras bibliotecas básicas.

Adicionalmente, entre las herramientas, *frameworks*, bibliotecas y demás aplicaciones de desarrollo que fueron utilizadas para la implementación del proyecto se encuentran las siguientes:

- **Semaphore:** Biblioteca que fue fundamental para la administración del recurso compartido. Esto en caso de que varios procesos pesados creados por el servidor intenten solicitar dos o más contraseñas concurrentemente.
- **OpenSSL/AES:** Biblioteca de código abierto que proporciona funciones criptográficas para realizar operaciones de cifrado y descifrado. En particular, .ªes.hçontiene las definiciones y funciones relacionadas con el algoritmo de cifrado AES (Advanced Encryption Standard). Útil para proteger la confidencialidad de los datos al permitir cifrar y descifrar información utilizando una clave secreta.
- **Socket:** Biblioteca que proporciona funciones, estructuras de datos y constantes para crear y manipular *sockets*, que funcionan como puntos finales para la comunicación de red.
- **Unistd:** Biblioteca que proporciona una amplia gama de funciones y constantes que son fundamentales para la programación del sistema y la gestión de procesos, como el `fork()`, `getpid()`, entre otras importantes para la creación de procesos pesados en el server.
- **Netinet:** Biblioteca que proporciona definiciones y estructuras relacionadas con la programación de redes, incluyendo la manipulación de direcciones IP y la configuración de *sockets* de red. Fue utilizada para implementar el protocolo UDP (User Datagram Protocol).
- **Types:** esta biblioteca define varios tipos de datos utilizados en las llamadas al sistema y otras interfaces del sistema. Incluye tipos como `pid_t` (ID de proceso).
- **Termios:** Utilizada para configuración y manipulación de las características del terminal. Fue de gran importancia en el desarrollo del *driver*.
- **Arduino:** Herramienta fundamentar para la interacción

con la placa Arduino UNO. Hizo posible el manejo del *hardware*.

- **Servo:** Biblioteca estándar que permite controlar servomotores utilizando la placa Arduino. Proporciona funciones y métodos para configurar y controlar los servomotores utilizados en la mano robótica de manera sencilla.
- **GitHub:** Plataforma que contiene el repositorio de la tarea. Fue de gran utilidad para el manejo de versiones, la sincronización y el acceso del código actualizado para los miembros del equipo.
- **Visual Studio Code:** Editor de texto que fue útil en el manejo y programación de los diferentes archivos de código en C, *makefile*, entre otros.

III. Detalles del diseño

A continuación, se presentan un conjunto de diagramas que ayudan a describir el funcionamiento general del sistema.

III-A. Diagrama de Funcionalidad

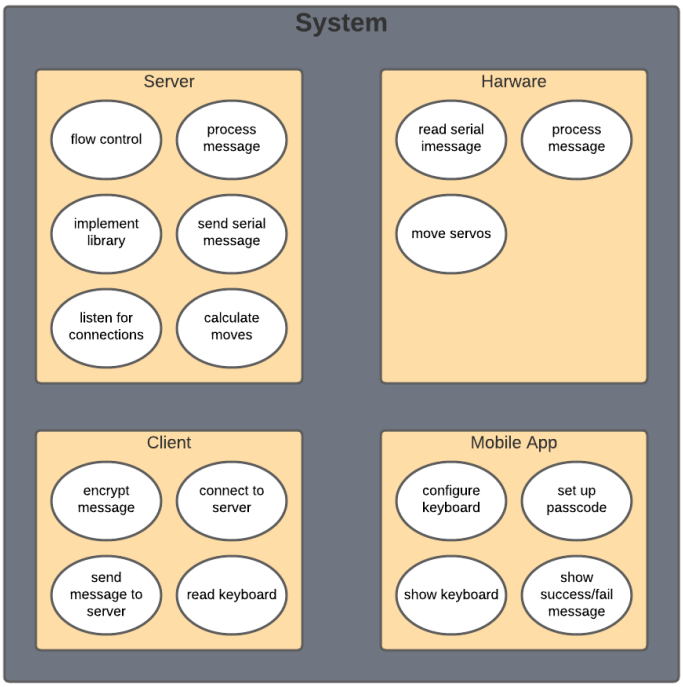

Figura 1. Diagrama de funcionalidad

En el fig 1 se pueden observar las funcionalidades más importantes de los componentes principales del sistema (Servidor, cliente, hardware y la aplicación móvil).

III-B. Diagrama de Arquitectura

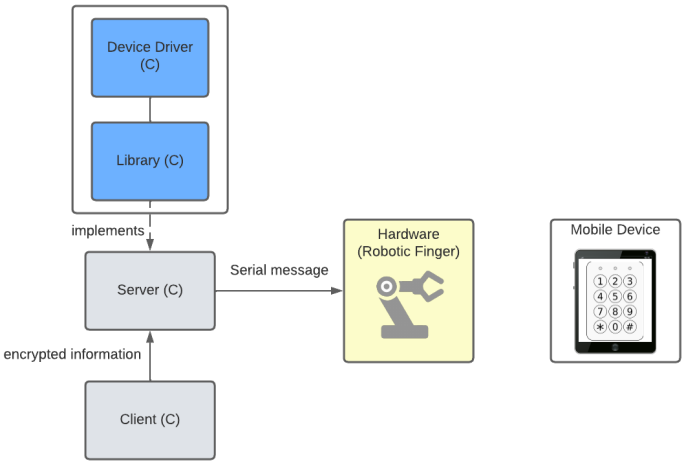

Figura 2. Diagrama de arquitectura de la solución planteada.

Tal y como se puede ver en la figura 2, el diagrama de arquitectura muestra las partes del sistema y la manera en la que se comunican o los subsistemas que implementan. En este diagrama se puede notar que el harware recibe un mensaje serial a través del servidor (El cual implementa la librería con las funciones especializadas del Device Driver), además, el cliente es el encargado de enviar un mensaje cifrado al servidor conteniendo las instrucciones que el dedo robótico debe interpretar. Finalmente, se puede ver que la aplicación móvil no está conectada a los demás componentes ya que no es una interacción directa del sistema.

III-C. Diagrama de Componentes

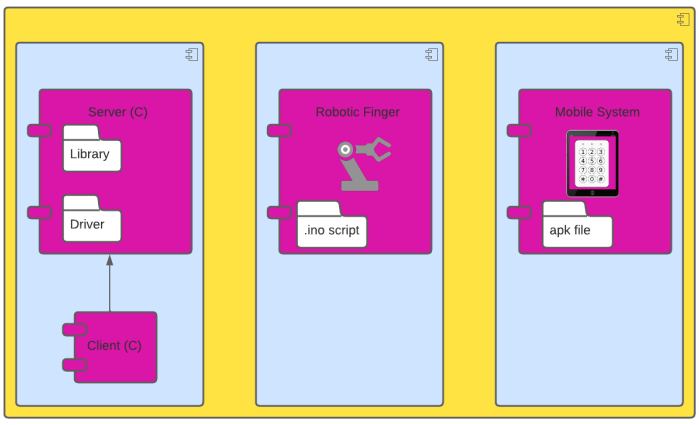

Figura 3. Diagrama de componentes de la solución planteada.

En la figura 3 se pueden apreciar los componentes principales del sistema, los cuales son el servidor, cliente, hardware y la aplicación móvil.

El servidor está conformado por una librería que implementa las funciones especializadas del driver, y toda la lógica de mensajes hacia el hadware o desde un cliente que haya sido aceptado.

El cliente contiene la lógica para establecer una conexión con el servidor, leer la entrada del usuario por el teclado y encriptar los mensajes.

El Hardware está compuesto por el prototipo del dedo robótico utilizando un Arduino y por supuesto, el script de Arduino con las instrucciones para interpretar los mensajes seriales y controlar el Hardware.

Por último, la aplicación móvil que está conformada por el dispositivo móvil (Una tablet) y el apk que se le instala para configurar y mostrar los teclados numéricos y los passcodes.

III-D. Diagrama de secuencia

A continuación, se presenta un diagrama de secuencia que describe el proceso de comunicación entre los componentes principales del sistema:

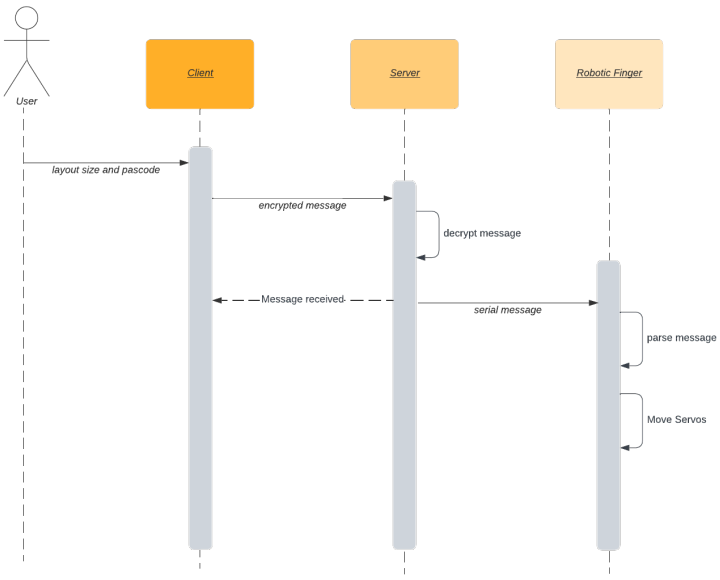

Figura 4. Diagrama de secuencia para describir el proceso de comunicación principal.

En la figura 4 se describe de manera general la ejecución del sistema para poder ingresar un código numérico en el dispositivo móvil:

1. El usuario ingresa un mensaje en la terminal activa del Cliente especificando el tamaño del teclado y el código numérico.
2. El Cliente envía el mensaje cifrado al servidor.
3. El Servidor descifra el mensaje.
4. El Servidor notifica el Cliente y envía un mensaje serial al Hardware con instrucciones más precisas para los movimientos y el tamaño del teclado .
5. El Hardware interpreta el mensaje serial.
6. El Hardware mueve los servos para ingresar el código numérico en el dispositivo móvil y luego presionar el botón de Submit para verificar si el código es correcto.

IV. Instrucciones de utilización del proyecto

Para la utilización adecuada del sistema, se debe hacer uso de las facilidades brindadas por parte de los desarrolladores para ejecutar cada apartado del sistema. De modo que se deben seguir los siguientes pasos:

- **Inicializar Driver:** El driver es la pieza principal de todo este sistema, puesto que es el que genera un ”puente.^{en}tre el apartado software y el apartado de hardware. De modo que, para que este pueda ser .^{a}ctivado”(Es decir, insertar el módulo compilado en el espacio del Kernel), el usuario debe dirigirse a la carpeta ”Driverz ejecutar el comando ”make”, el cual se encarga de los aparados de compilado y copia el módulo.
- **Inicializar Biblioteca:** La librería actuará como aquel apartado de software interactuará con el dispositivo USB mediante las facilidades que brindó la creación del driver. Para cargar la biblioteca a las bibliotecas del sistema, el usuario únicamente debe dirigirse a la carpeta ”Libraryz ejecutar el comando ”make”, el cual se encarga de ejecutar la secuencia de acciones necesaria para que el dicha biblioteca pueda ser utilizada como una biblioteca propia del sistema.
- **Inicializar Servidor:** El servidor es el encargado de ejecutar aquella lógica necesaria para desencriptar y comprender los mensajes enviados por el cliente, así como también generar un mensaje comprensible por el sistema de hardware creado, mediante la utilización de la biblioteca cargada anteriormente. Para inicializar este en la raíz de la carpeta principal del sistema, se debe ejecutar el comando ”make server”.
- **Conexiones:** Antes de proceder con la ejecución como tal, el usuario se debe cerciorar que las conexiones fueron efectuadas de forma correcta, es decir, el cable del Arduino debe estar conectado a un puerto del computador en el cual se encuentra el apartado de hardware ejecutado, y al mismo tiempo, el otro extremo del cable debe estar conectado al Arduino, cuyas conexiones con el dedo robótico se sobreentiende están conectadas de la misma forma en la que el equipo de desarrollo lo proveyó.
- **Inicialización de la Pantalla de Desbloqueo:** Una vez descargada e instalada la aplicación que proveyó el equipo de desarrollo, se muestra una pantalla inicial, esta le solicita al usuario la selección de una dimensión de pantalla, así como también la selección de un código el cual será el código de desbloqueo para el dispositivo.
- **Inicializar Cliente:** La labor del cliente es únicamente trasmitir un mensaje encriptado al servidor. Para inicializar el mismo, al igual que el servidor, en la raíz del folder

del sistema, se debe ejecutar el comando ”make client”. Una vez este este cargado, el usuario puede ingresar un código con el fin de que el dedo (El apartado de hardware creado para desbloquear el dispositivo), el mismo debe tener como primer valor numérico un número del uno al tres, este será el identificador de la dimensión de la pantalla de desbloqueo, siendo la dimensión grande el número uno, la dimensión media la número dos, y la dimensión pequeña el número tres, esto seguido del código deseado. Por ejemplo: 11234, indica que para el tamaño de pantalla grande, inserte el código 1234.

- **Ciclo de ejecución:** Una vez realizados todos estos pasos el sistema estará funcionando de forma ideal, y siguiendo toda aquella indicación que el usuario realice mediante el servidor, si el código y tamaño de pantalla seleccionados por el usuario en el apartado del cliente son los correctos, el dedo robótico se encargará automáticamente de desbloquear el dispositivo, y mostrar que éste fue desbloqueado de forma correcta, por el contrario, mostrará en pantalla que el código ingresado es incorrecto, y solicitará un nuevo intento.

Cabe destacar que la forma ideal de ejecutar el sistema en conjunto es siguiendo los pasos descritos anteriormente de forma secuencial.

V. Tabla de actividades por cada estudiante

A continuación se presenta la distribución de las tareas identificadas en la tarea por estudiante, así como datos como su fecha límite establecida, horas dedicadas y estado de completitud.

Descripción	Responsable (s)	Horas de trabajo	Estado
Implementación del cliente-servidor UDP	Julián	5	Completado
Implementación del método de cifrado	Julián	1	Completado
Implementación inicial del driver	Agustín	7	Completado
Sincronización en el servidor por medio de semáforos	Julián	1	Completado
Diseño del dispositivo de hardware para el dedo	Juan	4	Completado
Implementación de la matriz de números en la aplicación móvil	Kevin	8	Completado
Generación de la biblioteca del driver	Agustín	3	Completado
Elección del tamaño de la matriz en la aplicación	Kevin	2	Completado
Compra e instalación de los componentes de hardware	Juan	6	Completado
Creación de procesos pesados en el server para solicitudes	Julián	0.5	Completado
Verificación del código en la aplicación	Kevin	2	Completado
Ajuste del movimiento del hardware	Juan y Kevin	12	Completado
Integración de la biblioteca con el cliente-servidor	Julián y Agustín	3	Completado
Integración de todas las capas del proyecto	Todos	4	Completado
Redacción de documentación	Todos	4	Completado

Cuadro I
Tabla de actividades por estudiante

VI. Conclusiones

En el proceso de realización del trabajo fue posible comprobar la importancia de las diversas herramientas de trabajo y tecnologías que están a disposición para el implementación de este tipo de proyectos, ya que apoyan y facilitan las labores de los desarrolladores.

Adicionalmente, se reforzaron conceptos relacionados con el análisis de requerimientos, diseño de sistemas, desarrollo de programas en lenguaje C y utilización de protocolos de comunicación en procesamiento distribuido, al implementar efectivamente la arquitectura cliente-servidor por medio del protocolo UDP.

De igual manera, se reforzó el tema visto en el curso sobre procesos pesados al crear uno por cada solicitud de desbloqueo. Asimismo, la implementación exitosa de un método de cifrado de información entre el cliente y el servidor, ejemplificó la importancia de la seguridad en este tipo de sistemas como se explicó en el curso.

Al implementar el *driver* que proporcionó a las capas superiores las funciones necesarias para la interacción con el dispositivo físico, se concluye que integrar este tipo de módulos al *kernel* de Linux hace posible la escritura hacia el dispositivo de *hardware* para la especificación de directivas, así como la lectura de respuestas por parte del mismo. Este funcionamiento permitió el movimiento correcto del dedo robótico a partir de las indicaciones brindadas por el servidor a través de la biblioteca creada.

Adicionalmente, al implementar la capa de *hardware* fue posible reforzar conocimientos de diseño de sistemas de este tipo, que incluyen componentes como servomotores y mecanismos que hacen viable el movimiento en tres dimensiones.

Finalmente, durante el proyecto fue posible verificar el correcto funcionamiento del flujo completo del sistema, iniciando desde la capa de *software* con el cliente-servidor, la biblioteca y la aplicación móvil; hasta la capa *hardware* que corresponde al dedo robótico.

VII. Sugerencias y recomendaciones

El equipo de trabajo reflexionó sobre todo lo realizado, y se considera que en base a nuestras experiencias, podríamos aportar cierto conjunto de sugerencias y recomendaciones, las cuales serían muy útiles a futuro, para quien interese y desee entender y/o desarrollar un proyecto como el realizado:

- Si se trabajará en equipo, se recomienda mantener actividades, reglas y medidas que contribuyan a la compenetración del equipo, al desarrollo constante y al interés

por el trabajo individual de cada uno, esto contribuirá al entendimiento general de la aplicación y su desarrollo.
- Investigar y evaluar diferentes opciones de controladores de dispositivo para garantizar una compatibilidad adecuada entre el hardware del dedo robótico y el sistema operativo.
- Asegurarse de que todas las capas de software necesarias estén correctamente integradas y funcionando en armonía. Realizar pruebas exhaustivas para verificar la comunicación efectiva entre el hardware y el sistema operativo.
- Realizar pruebas exhaustivas del controlador de dispositivo para garantizar su correcto funcionamiento y compatibilidad con el hardware del dedo robótico
- Implementar mecanismos adecuados de sincronización (Semaforos) para garantizar una interacción fluida entre las diferentes partes del dedo robótico y evitar posibles problemas de concurrencia.

VIII. REFERENCIAS:

[1] Tanenbaum. Operating systems: design and implementation. Prentice Hall. 1988

[2] Peterson. Operating Systems Concepts. Addison Wesley, Second Edition. 1985

[3] Lubomir y Shaw, The logical design of operating systems. Prentice Hall, Second Edition. 1988.

[4] Kamburugamuve, S., Wickramasinghe, P., Govindarajan, K., Uyar, A., Gunduz, G., Abeykoon, V., Fox, G. (2018, July). Twister: Netcommunication library for big data processing in hpc and cloud environments. In 2018 IEEE 11th International Conference on Cloud Computing (CLOUD) (pp. 383-391). IEEE.

[5] Liguori, A. N., Wilson, M. S., Nowland, I. P. (2018). U.S. Patent No. 9,886,297. Washington, DC: U.S. Patent and Trademark Office.