Proyecto 2 – Proyecto libre

Kendall Martínez Carvajal - email: kendallmc@estudiantec.cr
Cristopher Eduardo Moreira Quirós - email: cris.moreira@estudiantec.cr
Irene Muñoz Castro – email: irmunoz@estudiantec.cr
Abraham Venegas Mayorga – email: abravenegas@estudiantec.cr
Gustavo Zamora Espinoza – email: guzamora@estudiantec.cr
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

***Abstract* – This project consists of developing a system that combines hardware and software to process information efficiently by using a distributed system. It is divided into four main phases: the design of the physical prototype based on GPIO interfaces, the creation of a custom device driver for communication with the hardware, the development of a specific library to facilitate the manipulation of the device and the implementation of a server-cluster that distributes the processing load among multiple nodes. The system encrypts the file before sending it to the cluster, which adds an extra layer of security. The cluster receives the encrypted file, decrypts it and processes it, identifying the most frequent word, which is then sent to a writing surface. All this in a Linux environment using communication through OpenMPI for the cluster.**

***Keywords*– clúster, driver, GPIO, kernel, procesamiento distribuido, UNIX.**

I. INTRODUCCIÓN

Los sistemas operativos desempeñan un rol crucial al actuar como un puente entre el hardware y el software, permitiendo que múltiples capas de programación interactúen eficazmente con dispositivos físicos. En este proyecto, se aplican conceptos avanzados de sistemas operativos, tales como la creación de drivers personalizados, el uso de hardware y la implementación de procesamiento distribuido y cifrado de datos. La integración de estas técnicas permite que un sistema operativo coordine la interacción fluida entre el hardware y el software, creando un sistema eficiente que responda a las necesidades de procesamiento distribuido.

Para iniciar se repasarán algunos conceptos necesarios para generar el proyecto, iniciando por el medio en donde se va a trabajar, ya que el proyecto está intrínsicamente relacionado al sistema operativo que se utilizará, en este caso Linux.

Linux es un sistema operativo (SO) gratuito y de código abierto basado en UNIX, creado por Linus Torvalds en 1991. Su naturaleza flexible permite a los usuarios modificar y crear distribuciones que satisfagan diversas necesidades informáticas. [6]

La arquitectura Linux tiene dos espacios clave:

- Espacio del kernel: responsable de las operaciones principales del sistema operativo y proporciona servicios esenciales.
- Espacio de usuario: donde se ejecutan las aplicaciones de usuario. Aquí, las aplicaciones de usuario aprovechan los servicios proporcionados por el kernel. [6]

Una interfaz de llamada al sistema facilita las interacciones entre las aplicaciones del usuario y el kernel, como se aprecia en Figura 1.

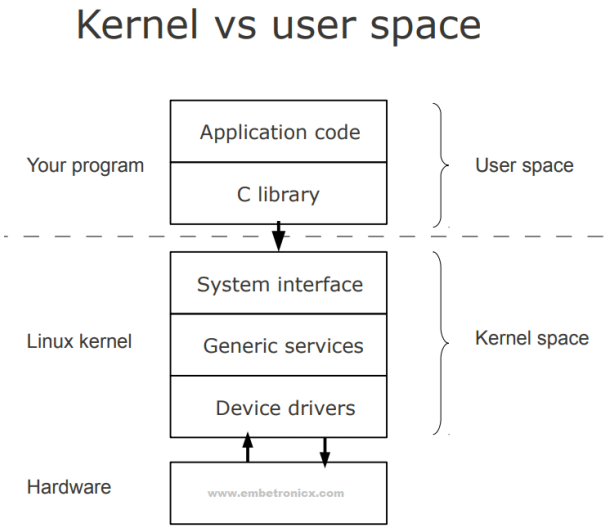

Figura 1: Diagrama de estructura de sistema arquitectura Linux, Kernel y espacio de usuario. [6]

En el desarrollo de controladores de Linux, la creación de un controlador de dispositivo generalmente implica la integración de código personalizado en el kernel. Hay dos formas de hacerlo. El enfoque tradicional implica la incorporación de código en el árbol de código fuente del kernel y la recopilación del kernel. Sin embargo, un método más eficiente implica la carga de código en el kernel

mientras está en funcionamiento, un proceso conocido como carga de módulos. [6]

Este método es más eficiente y flexible que el tradicional. En lugar de recompilar todo el kernel, los desarrolladores pueden crear módulos de kernel cargables (LKM) que contienen el código del controlador de dispositivo. Estos módulos se pueden cargar de forma dinámica en tiempo de ejecución sin necesidad de reiniciar el sistema. [6]

A pesar de su proceso de carga distintivo, los LKM son componentes integrales del kernel, que se comunican sin problemas con el núcleo base para cumplir con sus funciones designadas. Los LKM le permiten desarrollar, probar y mantener controladores de dispositivos con mayor facilidad, así como administrar los recursos del sistema con mayor flexibilidad. [6]

Al saber cómo está diseñado Linux y cuáles enfoques existen para generar drivers para dispositivos lo siguiente que se repasa es la forma en la que vamos a comunicar, la conexión del HW de la aplicación, en este caso por GPIO.

Los microcontroladores (MCU) se utilizan ampliamente para controlar dispositivos electrónicos de todo tipo, el MCU consta de una CPU (unidad central de procesamiento), memoria y circuitos adicionales que implementan una variedad de funciones de soporte de periféricos como se observa en Figura 2, lo que permite una implementación más sencilla en una variedad de configuraciones. Una MCU normalmente incluye una variedad de puertos de E/S (entrada y salida), por ejemplo, para facilitar el flujo de señales entre la CPU y los sensores y conmutadores externos. También suele incluir uno o más ADC (convertidores analógicos/digitales) para convertir las señales analógicas entrantes en valores digitales, y uno o más DAC (convertidores digitales/analógicos) para convertir los valores digitales en señales analógicas de salida. Estos puertos de E/S y convertidores permiten el uso de una variedad de tipos de señales. [3]

Un puerto GPIO (entrada/salida de propósito general) maneja señales digitales entrantes y salientes. Como puerto de entrada, se puede utilizar para comunicar a la CPU las señales de encendido/apagado recibidas de los interruptores o las lecturas digitales recibidas de los sensores. Como puerto de salida, se puede utilizar para controlar operaciones externas basadas en instrucciones de la CPU y resultados de cálculos. [3]

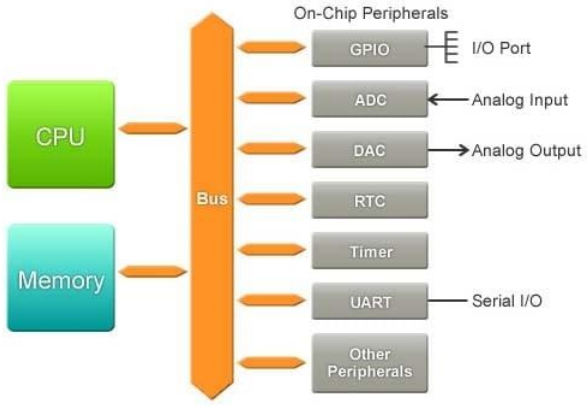

Figura 2: Configuración interna de la MCU (conceptual). [3]

El GPIO se conoce como "de propósito general" porque cada pin se puede configurar libremente para funcionar como entrada o como salida. En los primeros MCU, cada puerto era exclusivamente de entrada o exclusivamente de salida. Sin embargo, un GPIO es flexible. Si tiene 8 pines, puede configurarlos como mejor se adapte a sus necesidades: 4 de entrada y 4 de salida, o 7 de entrada y 1 de salida, o cualquier otra combinación. [3]

Tenga en cuenta que, si bien los programas leen, escriben y operan con valores digitales (0 y 1), los dispositivos externos suelen utilizar niveles de señal: voltaje BAJO y voltaje ALTO. El GPIO maneja las conversiones necesarias en ambas direcciones. [3]

Al momento de transmitir información existe la posibilidad de que esta se pierda o sea interceptada en el camino, por lo que al tomar en cuenta la seguridad se pueden ver diferentes maneras para abordar este problema.

El cifrado en ciberseguridad consiste en transformar los datos de un formato legible a uno codificado. Los datos cifrados únicamente pueden leerse o procesarse cuando se descifran. [4]

El cifrado es el pilar fundamental de la seguridad de los datos, es la forma más sencilla e inteligente de garantizar que la información de un sistema informático no pueda ser robada y leída por alguien con malas intenciones. Generalmente, los usuarios y las grandes empresas recurren al cifrado de seguridad de datos para proteger la información del usuario que se envía entre un navegador y un servidor. Dicha información incluye desde los datos de pago de un usuario hasta su información personal. El software de cifrado de datos, también conocido como algoritmo de cifrado o encriptación, se emplea para desarrollar un sistema de codificación. [4]

El cifrado consiste en convertir un archivo de texto legible en un texto incomprensible, implica modificar datos legibles de forma aleatoria. El cifrado requiere una clave

criptográfica y un conjunto de valores matemáticos acordados por el emisor y el destinatario. El destinatario descifra los datos con la clave, transformándolos de nuevo en un archivo de texto legible. [4]

Cuanto más compleja sea la clave criptográfica, más seguro será el cifrado. Así, será menos probable que terceros puedan descifrarla mediante ataques de fuerza bruta. El cifrado también se utiliza para proteger contraseñas. Los métodos de cifrado de contraseñas utilizan la codificación para que los hackers no puedan leerlas. [4]

Los dos métodos de cifrado más comunes son el cifrado simétrico y asimétrico. Los nombres indican si se utiliza o no la misma clave para codificar y descodificar:

- Claves de criptografía simétrica: También conocidas como cifrado de clave privada. La clave que se utiliza para el cifrado es la misma que la de descodificación, lo que facilita el proceso a los usuarios y a los sistemas cerrados. En caso contrario, la clave debe enviarse al destinatario. Esto puede ser más peligroso si la consigue un tercero. Este método es más rápido que el asimétrico. [4]
- Criptografía asimétrica: este método se vale de dos claves diferentes, públicas y privadas, que están relacionadas matemáticamente. Las claves son básicamente números grandes que se han emparejado entre sí sin ser idénticos, de ahí el uso del término asimétrico. El propietario guarda la clave privada en secreto y la clave pública se comparte entre los destinatarios autorizados o se pone a disposición del público en general. [4]

El siguiente punto que resaltar es buscar mejorar el rendimiento al procesar datos, en esta ocasión se reflejará usando un clúster con un sistema distribuido el cual dividirá las tareas para optimizar el procesamiento de la aplicación.

Un sistema distribuido es un conjunto de equipos independientes que actúan de forma transparente actuando como un único equipo. Su objetivo es descentralizar tanto el almacenamiento de la información como el procesamiento. [7]

Algunas de las ventajas que aporta un sistema distribuido son:

- Mayor eficacia.
- Mayor tolerancia a fallos: al estar distribuida la información en nodos, en caso de que se caiga un nodo, dicha información va a encontrarse replicada en otros nodos.
- Mayor velocidad y procesamiento distribuido: cuando se realiza una consulta, lo procesamientos se dividen entre todos los nodos que forman el sistema distribuido, en lugar de enviarlos a un único nodo y que el mismo tenga que hacer todo el trabajo.
- Escalabilidad: si, por ejemplo, se necesita más procesamiento o añadir más disco duro, en lugar de que los equipos crezcan de forma vertical añadiendo más almacenamiento, RAM o CPU, se añaden equipos de forma horizontal al clúster o sistema distribuido.[7]

Los clústeres son grupos de servidores que se gestionan en conjunto y que participan en la gestión de las cargas de trabajo. Un clúster puede contener nodos o servidores de aplicaciones individuales. Un nodo corresponde generalmente a un sistema físico que posee una dirección de host IP exclusiva y que ejecuta uno o más servidores de aplicaciones. Los clústeres pueden agruparse bajo la configuración de una celda. [5]

Los clústeres son responsables del equilibro del flujo de trabajo entre servidores. Los servidores que forman parte de un clúster se denominan miembros del clúster. Cuando instala una aplicación en un clúster, la aplicación se instala automáticamente en cada miembro del clúster. [5]

Para poder enviar la información de manera ordenada entre los nodos hay diferentes protocolos para envío de la información, nos centraremos en el uso de OpenMPI.

OpenMPI es una bilioteca que funciona como interfaz de paso de mensajes de código abierto desarrollada y mantenida por un conjunto de socios académicos, de investigación e industriales. MPI (Message Passing Interface) se utiliza principalmente en computación paralela distribuida. Es un protocolo de comunicación utilizada para la computación paralela. [10]

Para llevar a cabo el proyecto, se divide en cuatro fases. La primera fase implica el diseño de un prototipo físico que pueda interactuar con el entorno mediante interfaces GPIO. La segunda fase se centra en el desarrollo del driver, que no solo permitirá la comunicación con el dispositivo, sino también servirá como una base sobre la cual el sistema operativo pueda identificar y controlar el hardware. La tercera fase corresponde a la creación de una biblioteca que contenga funciones específicas para manipular el dispositivo de manera intuitiva. Finalmente, la cuarta fase consiste en la configuración del servidor-clúster, que distribuye la carga de procesamiento entre los nodos, recibe el archivo cifrado del cliente y, una vez que el archivo es descifrado y procesado, envía la palabra identificada como más frecuente a la superficie de escritura.

II. AMBIENTE DE DESARROLLO

Para garantizar la correcta ejecución del proyecto, se requiere una configuración básica del entorno de desarrollo que incluya frameworks, bibliotecas externas y

principales, aplicaciones de terceros y herramientas de desarrollo. A continuación, se describen los elementos utilizados:

Sistema Operativo: El proyecto se desarrolla en un entorno **Linux**, específicamente en una distribución basada en **Ubuntu 22.04 LTS**, debido a su compatibilidad con herramientas de desarrollo y manejo eficiente de drivers personalizados.

Frameworks y bibliotecas principales:

- **Arduino IDE:** Para la programación y carga de scripts en la placa Arduino UNO.
- **SSH (Secure Shell):** Utilizada para la comunicación segura entre los nodos del clúster, permitiendo la ejecución remota de comandos y transferencia de datos.
- **OpenMPI:** Herramienta fundamental para el procesamiento distribuido en el clúster, facilitando la comunicación entre nodos y la división eficiente de las tareas.
- **Net-tools:** Conjunto de utilidades de red para la configuración y monitoreo de las interfaces y conexiones de red del clúster.

Aplicaciones de terceros:

- **Git:** Utilizado para el control de versiones del código fuente.
- **SharePoint y Google Drive:** Para la colaboración en la documentación del proyecto y el intercambio de archivos entre los miembros del equipo.
- **Make:** Herramienta de automatización para compilar y gestionar dependencias del proyecto.

Herramientas de desarrollo:

- **gcc:** Compilador de C para compilar el cliente-servidor y los módulos del kernel.
- **VS Code:** Entorno de desarrollo integrado (IDE) para edición de código, con extensiones para C y Arduino.

III. ATRIBUTOS

A. Aprendizaje continuo

• ¿Cuáles son las necesidades actuales de aprendizaje para enfrentar el proyecto?

Investigación sobre la generación de drivers específicos para la aplicación del proyecto, así como la investigación sobre cómo generar el clúster de manera que la información se procese de manera óptima.

• ¿Cuáles son las tecnologías que se pueden utilizar para el desarrollo?

Entre las tecnologías que se pueden implementar están los lenguajes de programación como C para la lógica del clúster, el driver y cifrado de los datos por ejemplo y arduino para implementar la lógica del microcontrolador.

Las herramientas de desarrollo como Arduino IDE para la programación de la placa Arduino, y gcc para compilar el código en C son herramientas indispensables para el desarrollo del proyecto.

Los frameworks y bibliotecas son también tecnologías que se pueden utilizar para el desarrollo del proyecto.

Por otro lado, una Plataforma colaborativa como GitHub que también funciona como plataforma de control de versiones es una herramienta que se puede utilizar para el desarrollo del proyecto y que además facilita muchos aspectos técnicos y de comunicación.

• ¿Cuáles acciones se implementó para el desarrollo del proyecto (organización de tiempo, búsqueda de información, repaso de contenidos, entre otros)?

Se utilizó un diagrama de Gantt para visualizar las tareas con respecto al tiempo necesario para invertir. El cual podemos ver en la Figura 3.

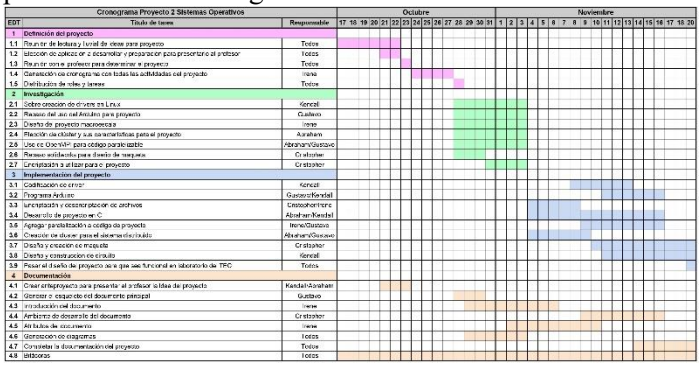

Figura 3. Diagrama de Gantt.

• Evalué de forma crítica la eficiencia de las acciones implementadas en el contexto tecnológico.

Para algunas partes como lo sería el cifrado de los datos para mantener la seguridad de la información, lo estamos realizando de manera simétrica con el algoritmo más sencillo el cual puede ser fácilmente descifrado, sin embargo, conocemos de otros tipos que son mucho más seguros, pero por cuestión de complejidad del proyecto y tiempo que maneja el equipo, se tomaron estas decisiones. Lo que sí es acertado es cifrar la data de alguna forma para que al menos no sea permanentemente visible.

Otra de las acciones es el hecho de usar un clúster para procesar la información, ya que esto es algo que se utiliza mucho en el contexto actual, ya que como se menciona en

la introducción, tiene muchas ventajas como lo es la escalabilidad, mejora en la velocidad y tolerancia a fallos.

B. Trabajo individual y en equipo

• Indicar las estrategias para el trabajo individual y en equipo de forma equitativa e inclusiva en las etapas del proyecto (planificación, ejecución y evaluación).

Para asegurar un trabajo equitativo e inclusivo, cada miembro del equipo busca entre sus conocimientos técnicos y fortalezas para encontrar cuál puede ser la manera para aportar al proyecto. En la etapa de planificación, se distribuyen las tareas para cada miembro y se tiene una lluvia de ideas para elegir entre todos la aplicación, tomando en cuenta las opiniones de todos los miembros del equipo dejando que puedan defender las ideas propuestas por cada integrante.

• Indicar la planificación del trabajo mediante la identificación de roles, metas y reglas.

En la planificación, cada miembro del equipo asumió roles específicos para cada fase del proyecto. Se definieron metas bisemanales para indicar el avance individual, así como el grupo que se tiene para comunicación constante en caso de que los integrantes necesiten ayuda sin necesidad de esperar a llegar a las reuniones principales. Las reglas del equipo incluyen, plazos establecidos para las diferentes tareas y la revisión constante de código entre pares (pair programming).

• Indicar cuales acciones promueven la colaboración entre los miembros del equipo durante el desarrollo del proyecto.

Para fomentar la colaboración, se establecieron reuniones de seguimiento. El equipo utilizó plataformas colaborativas tanto para codificación como lo es GitHub para el control de versiones, como para la documentación, por un sharepoint, donde todos los integrantes tienen acceso a los avances de los compañeros.

• Indicar cómo se ejecutan las estrategias planificadas para el logro de los objetivos.

Las estrategias planificadas se ejecutaron a través de la organización bisemanal de entregables parciales y pair programming permitiendo el avance constante. Principalmente nos basamos en la modulación del proyecto para no tener que dependen tanto de los compañeros y presentar los avances en los plazos asignados.

• Indicar la evaluación para el desempeño del trabajo individual y en equipo

El desempeño individual fue evaluado mediante la revisión de los entregables asignados a cada integrante, el cumplimiento de plazos y la resolución de problemas. A nivel de equipo, se midió el nivel de colaboración y la eficacia en la comunicación.

• Indicar la evaluación para las estrategias utilizadas de equidad e inclusión.

La equidad e inclusión fueron evaluadas tomando en cuenta la participación de todo el equipo en las decisiones y la distribución balanceada de las tareas. Evitando en la medida de lo posible la sobrecarga de trabajo y dando la oportunidad de que todos pudieran participar en actividades.

• Indicar la evaluación para las acciones de colaboración entre los miembros del equipo.

La colaboración es evaluada en función del éxito en la integración de las diferentes fases del proyecto y la capacidad de resolver problemas técnicos. Se consideró el grado en que cada miembro aportó para ayudar a solucionar problemas de otros compañeros, así como la participación en las reuniones y plataformas colaborativas.

IV. DETALLES DE DISEÑO

En esta sección se describen los componentes de software y hardware del proyecto, junto con los diagramas necesarios para ilustrar el funcionamiento del sistema.

A. Hardware

El hardware del sistema está compuesto por varios componentes integrados para cumplir con las funcionalidades propuestas:

1) Botones de entrada: El dispositivo cuenta con cuatro botones que van del 1 al 4. Estos botones son utilizados para que el usuario pueda ingresar una combinación con ellos y cada uno estará conectado a pines GPIO del arduino. La combinación ingresada es enviada mediante serial al cliente.

2) Buzzer: Este componente genera sonidos según el binario recibido por parte del servidor. Este también se encuentra controlado por pines GPIO para controlar dichos sonidos, en donde si se encuentra un 1 el buzzer emite sonido y si encuentra un 0, no emite sonido.

3) Arduino UNO: Este microcontrolador se encarga de gestionar la lógica del hardware y la comunicación con el cliente/servidor, esto con ayuda de la comunicación serial.

4) LEDs: Se tendrán dos de estos, uno rojo y uno verde, el rojo indica que está encendido y se encuentra en espera y el verde es que está intentando reproducir una secuencia.

Las conexiones del circuito completo se pueden observar en la Figura 4.

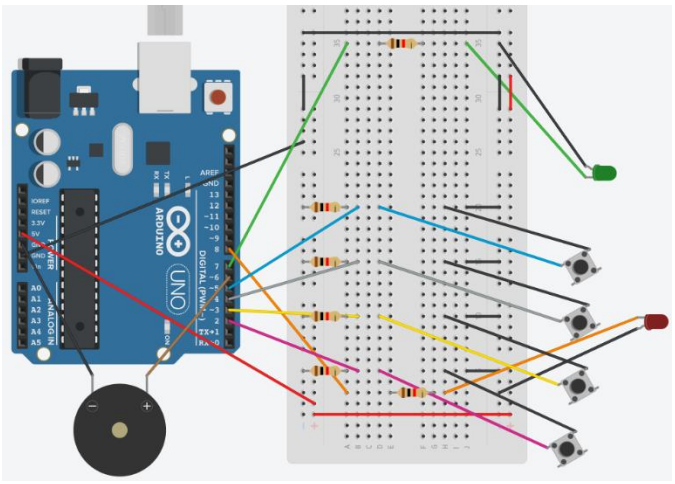

Figura 4. Diagrama del circuito.

B. Software

1) Captura de combinaciones: Se utiliza Arduino IDE para capturar las combinaciones que ingresa el usuario a través de los botones y esta es enviada por medio de serial a un programa en C que corresponde al cliente.

2) Cliente: Se utiliza el lenguaje de programación C para realizarlo. Aquí se recibe la combinación de los botones para luego ser cifrada y enviada al servidor mediante el uso de sockets.

3) Algoritmo de cifrado: Se utiliza el algoritmo AES para cifrar la información, de esta forma se asegura que los datos enviados estén protegidos correctamente.

4) Servidor/Clúster: Para esto se crean tres nodos para realizar el procesamiento distribuido, en donde la carga se distribuye de la siguiente manera:

- Nodo 1: Recibe la información cifrada y se encarga de descifrarla para luego pasarla al nodo 2.
- Nodo 2: Se encarga de buscar el archivo correspondiente a la combinación ingresada. Una vez que lo encuentra, toma el contenido que corresponde a un número decimal y lo convierte a su equivalente en binario, el cual se pasa al nodo 3.
- Nodo 3: Toma el número binario, lo pasa a una cadena de texto y lo envía al hardware mediante el uso de puertos serial.

C. Diagrama de secuencia

El diagrama de secuencia describe el proceso completo desde la generación de la combinación con los botones hasta la emisión de sonido. En la figura 5 se puede observar este proceso.

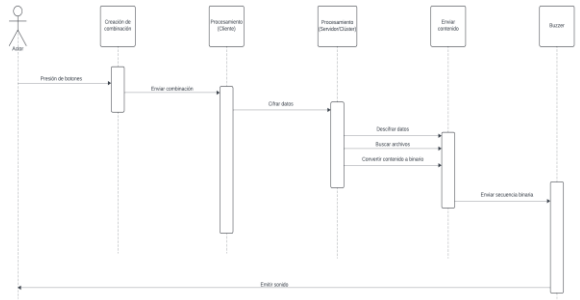

Figura 5. Diagrama de secuencia.

D. Diagrama de componentes

Este diagrama presenta los principales módulos del sistema y las relaciones entre ellos, esto se puede ver en la figura 6.

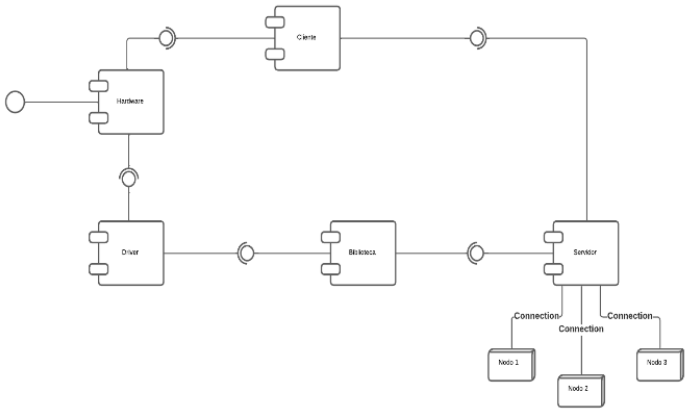

Figura 6. Diagrama de componentes

E. Diagrama de flujo

Este diagrama ilustra a grandes rasgos el proceso de ejecución del proyecto, esto se puede observar a continuación en la Figura 7.

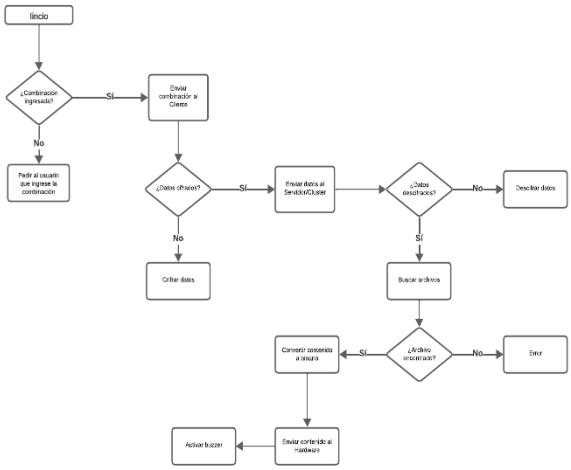

Figura 7. Diagrama de flujo

V. INSTRUCCIONES DE USO

Inicialización del cluster:

Preparación del sistema en los nodos (Master y Slave):

1. Asegúrate de que cada nodo tenga instalado **Ubuntu** (o una distribución similar) y que cuentes con acceso **sudo**.
2. Crea un usuario para el clúster en cada nodo:

```
>>sudo adduser mpiuser
>>sudo usermod -aG sudo mpiuser
```

3. Cambia al usuario creado:

```
>>su mpiuser
```

Instalación de dependencias:

4. Instala OpenMPI en cada nodo:

```
>>sudo apt-get install openmpi-bin
```

5. Instala el servidor SSH para comunicación entre los nodos:

```
>>sudo apt-get install openssh-server
```

Configuración de comunicación entre nodos:

6. Genera claves SSH en el nodo maestro (no uses frase de paso al generarlas):

```
>>ssh-keygen
```

7. Copia la clave pública a cada nodo esclavo:

```
>>ssh-copy-id mpiuser@<slave-ip-address>
```

8. Verifica la conexión SSH entre el maestro y los esclavos:

```
>>ssh mpiuser@<slave-ip-address>
```

Configuración del archivo de hosts:

9. En cada nodo, edita el archivo /etc/hosts para incluir las direcciones IP de los nodos maestro y esclavos. Ejemplo:

```
192.168.1.10    master
192.168.1.11    slave1
192.168.1.12    slave2
```

10. Guarda los cambios con `CTRL+O` y sal con `CTRL+X`.

Ejecución del programa MPI:

11. Compila el programa utilizando el comando `make` en el directorio del proyecto:

```
>>make
```

12. Ejecuta el cliente con el siguiente comando:

```
>>./Cliente
```

13. Ejecuta el programa MPI utilizando los nodos maestro y esclavos definidos:

```
>>mpiexec -n 3 -host
192.168.1.10,192.168.1.11,192.168.1.12
>>./mpi_test
```

Uso de la aplicación:

Encendido de la aplicación:

- Al iniciar, el **LED rojo** estará encendido, lo que indica que la aplicación está esperando que se ingrese una secuencia de números.

Ingreso de la secuencia:

- La secuencia debe ser de **4 números** y se ingresa presionando los botones del hardware.
- Cada botón tiene un valor asignado:
  - **Botón más a la izquierda:** Representa el número 4.
  - **Botón más a la derecha:** Representa el número 1.
- Los botones intermedios representan los números 3 y 2, de izquierda a derecha, respectivamente.

Confirmación de la secuencia:

- Una vez presionados los 4 botones correspondientes a la secuencia, el **LED rojo** se apagará y se encenderá el **LED verde**, indicando que la secuencia ha sido recibida correctamente.

Reproducción de la secuencia:

- Tras la confirmación, el hardware emitirá la secuencia ingresada en formato binario mediante el buzzer, reproduciendo cada valor según el orden en el que se ingresó.

VI. BITÁCORA

Esta bitácora resumen se encuentra en la Tabla I que está basada en las actividades planteadas del diagrama de Gantt

en la Figura 3. En actividades están los EDT de las actividades.

TABLA I
BITACORA DE LAS HORAS INVERTIDAS POR ESTUDIANTE EN EL PROYECTO

Estudiante	Actividades	Total horas trabajadas
Abraham	1.1, 1.2, 1.3, 1.5 2.4, 2.5, 3.4, 3.6 3.9, 4.1, 4.6, 4.7 4.8	57 horas
Kendall	1.1, 1.2, 1.3, 1.5 2.1, 3.1, 3.2, 3.4 3.8, 4.1, 4.6, 4.7 4.8	60 horas
Cristopher	1.1, 1.2, 1.3, 1.5 2.6, 2.7, 3.3, 3.7 3.9, 4.4, 4.6, 4.7 4.8	50 horas
Irene	1.1, 1.2, 1.3, 1.4 1.5, 2.3, 3.3, 3.5 3.9, 4.3, 4.5, 4.6 4.7, 4.8	50 horas
Gustavo	1.1, 1.2, 1.3, 1.5 2.2, 2.4, 3.2, 3.5 3.6, 3.9, 4.2, 4.6 4.7, 4.8	50 horas

VII. CONCLUSIONES

La creación de un clúster para distribuir tareas entre nodos garantiza un procesamiento optimizado, mostrando la escalabilidad y tolerancia a fallos como características clave del diseño.

Se concluyó que es posible diseñar e implementar un driver personalizado para Linux que permita la interacción eficiente con el hardware a través de interfaces GPIO, destacando la flexibilidad de este sistema operativo para tales tareas.

Se concluyó que el sistema distribuido, al replicar información entre nodos, es tolerante a fallos parciales, asegurando la continuidad del procesamiento incluso si un nodo deja de funcionar.

El proyecto permitió concluir que la división de tareas y roles claros ayudó a cumplir con los objetivos del proyecto dentro de los plazos establecidos, demostrando la importancia de una planificación y comunicación efectivas.

Se concluyó que es posible diseñar e implementar un driver personalizado para Linux que permita la interacción eficiente con el hardware a través de interfaces GPIO, destacando la flexibilidad de este sistema operativo para tales tareas.

VIII. SUGERENCIAS Y RECOMENDACIONES

Si bien el cifrado AES básico asegura un nivel inicial de seguridad, se sugiere explorar versiones más avanzadas del mismo algoritmo o adoptar métodos asimétricos que ofrezcan mayor robustez. Esto sería especialmente útil para aplicaciones en entornos más críticos.

Para futuros proyectos, se recomienda al equipo profundizar en temas como criptografía avanzada, sistemas distribuidos y optimización de controladores. Esto asegurará una capacidad técnica creciente para abordar proyectos de mayor complejidad.

Es recomendable la utilización de herramientas como JIRA o Asana pues podrían mejorar la planificación y el seguimiento de tareas, proporcionando métricas más detalladas sobre el rendimiento del equipo.

Implementar pruebas en entornos reales o simulados con diferentes niveles de carga permitirá evaluar la resiliencia y escalabilidad del sistema de manera más exhaustiva.

Se recomienda evaluar el consumo energético del sistema, especialmente en el hardware, para identificar posibles mejoras en términos de sostenibilidad.

Realizar pruebas de carga para evaluar el rendimiento del clúster bajo condiciones extremas garantizará que el sistema pueda soportar incrementos inesperados en el volumen de datos.

REFERENCIAS

[1] Javier. (2019). *How to write a linux device driver* [Online]. Available: https://jvgd.medium.com/how-to-write-a-device-driver-76d0584a4be3

[2] EmbeTronicX. (2023). *Linux Device Drivers tutorial | Linux drivers and kernel modules* [Online]. Available: https://embetronicx.com/tutorials/linux/device-drivers/linux-device-driver-part-1-introduction/

[3] Renesas. (2024). *Essentials of Microcontroller Use Learning about Peripherals: GPIO* [Online]. Available: https://www.renesas.com/en/support/engineer-school/mcu-programming-peripherals-01-gpio?srsltid=AfmBOoprN22DisSN61PpeQqGEMs6-EEuyy8RgXEPc2f3anBH9Glfe-RZ

[4] Kaspersky. (2018). *¿Qué es el cifrado de datos? Definición y explicación* [Online]. Available: https://www.kaspersky.es/resource-center/definitions/encryption

[5] IBM Corporation. (2024). *Introducción: clústeres* [Online]. Available: https://www.ibm.com/docs/es/was-zos/9.0.5?topic=servers-introduction-clusters

[6] Dmitriy. (2024). *Linux Device Drivers: Tutorial for Linux Driver Development* [Online]. Available: https://www.apriorit.com/dev-blog/195-simple-driver-for-linux-os

[7] Losada, S. (2018). *Qué es un sistema distribuido y qué ventajas aporta su funcionamiento* [Online] Available: https://openwebinars.net/blog/que-es-un-sistema-distribuido/

[8] Butler, S. (2022). *What Is GPIO, and What Can You Use It For?* [Online]. Available: https://www.howtogeek.com/787928/what-is-gpio/

[9] Nordic Semiconductor ASA .(2024). *GPIO — General purpose input/output* [Online] Available: https://docs.nordicsemi.com/bundle/ps_nrf9160/page/gpio.html

[10] Open MPI Project. (2024). *Open MPI: Open Source High Performance Computing* [Online]. Available: https://www.open-mpi.org/