REINGTEC

Revista de Ingeniería y Tecnologías para el Desarrollo Sustentable 12 (2024) 169 - 178

ISSN: 2448 - 7198

ANÁLISIS COMPARATIVO DE TRANSMISIÓN DE DATOS VIA PUERTO USB OTG PARA APLICACIONES DEL PLC

COMPARATIVE ANALYSIS OF DATA TRANSMISSION VIA USB OTG PORT FOR PLC'S APPLICATIONS

Negrete Medellín Arturo[1], Elizalde Canales Francisca Angélica[1], Rivas Cambero Iván de Jesús[1]

[1]Universidad Politécnica de Tulancingo, Universidad Politécnica de Tulancingo, Tulancingo de Bravo, Hgo. México 43629, *arturo.negrete@upt.edu.mx

RESUMEN. *El presente documento aborda el desarrollo de software para la implementación de un puerto USB-OTG (Universal Serial Bus On The Go) y la comparación en diferentes situaciones de una transmisión de datos de alta velocidad en una tarjeta de desarrollo que haga las funciones de un Controlador Lógico Programable (PLC por sus siglas en ingles programmable logic controller). Se presenta una metodología que incluye la evaluación de requisitos, la selección de la arquitectura del software, la implementación y validación de la conexión USB-OTG para obtener una opción más de comunicación para el PLC tan necesaria en esta era de la industria 4.0. Los resultados muestran mejoras en la velocidad de comunicación con respecto a varios factores, contribuyendo a una mejora en la eficiencia del control y monitoreo de procesos industriales. Las conclusiones destacan los factores que influyen de manera significativa en la transmisión de datos a través del puerto USB-OTG para su operación efectiva y segura en el ciclo de escaneo del PLC.*

Palabras clave: *Conexión, ciclo, escaneo*

ABSTRACT. *This document addresses the development of software for the implementation of a USB-OTG (Universal Serial Bus On The Go) port and the comparison in different situations of a high-speed data transmission on a development card that acts as a Programmable Logic Controller (PLC). A methodology is presented that includes the evaluation of requirements, the selection of the software architecture, the implementation and validation of the USB-OTG connection to obtain one more communication option for the PLC so necessary in this era of the industry 4.0. The results show improvements in the communication speed with respect to several factors, contributing to an improvement in the efficiency of control and monitoring of industrial processes. The conclusions highlight the factors that significantly influence data transmission through the USB-OTG port for its effective and secure operation in the PLC scan cycle..*

Key words*: Connection, cycle, scanning*

INTRODUCCIÓN

En el mundo de la tecnología, tanto los PLC como los USB desempeñan roles cruciales, cada uno en su propio dominio: El PLC como cerebro de la automatización industrial [1], los PLC son computadoras robustas diseñadas para ejecutar tareas repetitivas y tomar decisiones basadas en datos de sensores y otros dispositivos, de diseño flexible y fácil de modificar [2]. La tecnología USB es un estándar de conexión que facilita la comunicación entre dispositivos electrónicos [3]. Su simplicidad y flexibilidad lo han convertido en una interfaz omnipresente en la computación moderna.

En el ámbito industrial, la conexión USB ofrece nuevas posibilidades para la comunicación entre PLC y computadoras. Al aprovechar la alta velocidad de transferencia y la amplia gama de dispositivos USB disponibles [4], se pueden implementar soluciones más flexibles y eficientes para la adquisición de datos, el control de procesos y la monitorización de sistemas.

La conexión USB directa entre dos computadoras ofrece una alternativa a las redes Wi-Fi [5], especialmente cuando se prioriza la velocidad en transferencias de datos. Sin embargo, existen limitaciones técnicas que impiden conectar dos anfitriones USB de forma nativa. Para superar esta restricción, se utilizan cables con chips que simulan otras interfaces, aunque a costa de una menor velocidad de transferencia.

La migración hacia el USB y Ethernet/IP en los PLC, como se observa en los productos de Allen-Bradley, representa una oportunidad para explorar nuevas aplicaciones y funcionalidades. Esta tendencia refleja

la búsqueda constante de soluciones más eficientes y versátiles en la automatización industrial. Se presenta una revisión de las investigaciones más recientes en control industrial, centrándonos en la tecnología USB OTG.

- Módulo Terminal Remoto, para la adquisición de datos, monitoreo y control de procesos Agroindustriales [6].
- Remote supervision and control based on wireless technology to operation of central pivot irrigation machine [7].
- Embedded Real Time Media Streaming over Ethernet via USB-OTG [8].
- USB OTG communication [9].

Resumiendo, este estudio tiene como objetivo explorar la viabilidad y los beneficios de integrar puertos USB-OTG en los PLC. Al hacerlo, se busca ofrecer una alternativa de comunicación segura y local, además de simplificar las tareas de programación y configuración de estos dispositivos industriales [10].

METODOLOGÍA

La metodología de este estudio se dividió en cuatro fases: la investigación, la preparación del entorno de trabajo, el diseño y desarrollo del software, y por último la validación y pruebas del sistema.

Investigación

Una implementación exitosa de USB OTG requiere una planificación minuciosa y la evaluación de una variedad de factores técnicos.

Diseño y desarrollo: La elección de la Raspberry Pi Zero 2W, junto con Raspbian y la API USB Gadget, resultó ideal para el diseño y desarrollo del proyecto [11]. Esta combinación ofrece una plataforma robusta y flexible para la implementación de USB OTG, gracias a su compatibilidad con las últimas actualizaciones de Linux-USB y a una interfaz de programación unificada.

Conexión: De los 3 cables USB-A a micro USB evaluados, solo 2 resultaron funcionales: uno incluido con un smartphone compatible con OTG (30 cm) y otro cable comercial (150 cm). Un cable de cargador (30 cm) no superó las pruebas.

Software: Se emplearon dos computadoras, una con Windows 10 y otra con Windows 11, ambas actualizadas. Estas máquinas utilizaron el mismo controlador USB Ethernet/RNDIS Gadget para reconocer la Raspberry Pi Zero 2W como una tarjeta de red Ethernet, optimizando así la conexión.

Pruebas funcionales: Se llevaron a cabo pruebas con diversas configuraciones de red (Ethernet, Wi-Fi y USB-OTG) utilizando diferentes cables para evaluar el desempeño de la conexión USB-OTG.

Seguridad: Se implementarán múltiples capas de seguridad. En primer lugar, se limitará el número de conexiones al servidor y se restringirá el acceso a direcciones IP específicas. Además, se establecerá una segmentación de red, aislando la conexión USB-OTG de la red principal. En escenarios de máxima seguridad, se restringirá el tráfico de internet a solo estados del PLC, mientras que el control total se realizará a través de USB-OTG. Finalmente, se implementará un sistema de autenticación basado en credenciales encriptadas y se definirán niveles de acceso para los comandos de control; otra manera de detener intrusos sería el realizar la conexión con clientes de diferente rango de direcciones TCP (Transmission Control Protocol) a una funcionalidad diferente que no permita que ocurran intromisiones en el proceso del PLC (aún no probado).

Actualizaciones: Al ser un derivado de Debian, el sistema operativo Raspberry Pi sigue un ciclo de lanzamiento bienal. Las imágenes ISO oficiales se encuentran en raspberrypi.com/software/operating-systems/ [12]. Se recomienda realizar actualizaciones periódicas para mantener el sistema seguro y actualizado. En el caso de Windows, es esencial instalar las actualizaciones críticas y actualizar anualmente el IDE de C++ Builder para aprovechar las últimas mejoras.

Métrica para las pruebas: Para evaluar la eficacia de la conexión USB, se comparó la velocidad de transferencia real con la velocidad teórica máxima establecida por el fabricante. Se consideró que el protocolo TCP garantiza la integridad de los datos

transmitidos. La eficacia se calculó como el cociente entre la velocidad medida y la velocidad ideal, expresando el resultado como un porcentaje.

La carga del ciclo de comunicación en un PLC representa la porción de la capacidad de procesamiento de la CPU dedicada a las tareas de comunicación [13]. El sistema operativo asigna de manera continua un porcentaje específico de esta capacidad al proceso de comunicación, empleando una técnica de intervalos de tiempo. Por lo tanto, hay que considerar la importancia del ciclo del PLC [14], centrado en la lectura de entradas, ejecución del programa y actualización de salidas [15], ya que limita los tiempos disponibles para comunicación. La velocidad de la CPU y la complejidad del programa [16] influyen directamente en estos tiempos.

Dado que las tendencias como la Industria 4.0 predicen el uso de control descentralizado y una mayor inteligencia [17, 18], se exigirá una mayor velocidad de proceso y de transmisión de datos, porque si en cada unidad de control, se podrán tomar decisiones, éstas deberán transmitirse a las unidades dependientes en tiempo y forma, tomando en cuenta que, cada decisión podría afectar otros procesos y al incorporar una opción más de comunicación como el USB-OTG en el PLC, se pretende cubrir las exigencias de comunicación a futuro.

De todas las opciones encontradas, cuatro fueron las tarjetas que tenían integrada una conexión USB OTG (On The Go) que permiten una conexión directa con cualquier computadora, las demás requieren de un circuito puente; se eligen la tarjeta Raspberry Pi Zero 2W como servidor para realizar las pruebas de software y de conexión por sus características ideales para este trabajo; las otras dos opciones con USB-OTG integrado son Raspberry Pi Zero W, Raspberry Pi 4.0 B+ y la ESP32-S3-USB-OTG por falta de disponibilidad o tener similitud con la tarjeta seleccionada, no se utilizaron para las pruebas.

Además, con el objetivo de obtener más resultados, se utilizaron dos computadoras personales PC con procesadores Intel i3 con 4 GB de RAM e i5 Gen. 11 con 8 GB de RAM con sistemas Ubuntu 22.04 con el servidor y Windows 11 con el cliente respectivamente.

Otro de los puntos importantes dentro del proyecto, es la selección del lenguaje de programación ya que será el pilar de todo el trabajo que se va a realizar tomando en cuenta

- Energía de consumo
- Tiempo de ejecución
- Memoria
- Dominio en el lenguaje
- Tiempo de desarrollo
- Mantenimiento

Un estudio realizado por investigadores portugueses ya trataba de dar respuesta a esa pregunta en 2017. En su análisis consideraron tanto el uso de memoria como la energía consumida en la ejecución de un conjunto de programas especiales. El conjunto de programas llamado Computer Language Benchmarks Game es un proyecto de Software Libre que permite comparar cómo ciertos algoritmos se implementan en diversos lenguajes de programación. En ese proceso se analizan el tiempo de ejecución, el consumo de memoria y el gasto de energía. En el análisis se compararon distintos lenguajes de programación que pertenecen a distintos paradigmas: funcionales, imperativos, orientados a objetos y de scripting. Considerando desde lenguajes clásicos como C o C++ a alternativas más modernas como Rust y también muy conocidas como Javascript, Perl, PHP o Python. [19].

Todo lenguaje tiene un propósito por el cual fue creado y, dependiendo de las necesidades que tenga un proyecto, se tendrá que desarrollar en un lenguaje u otro. Por ello, es muy importante que todo proyecto tenga una fase de planificación donde se haga un estudio de las necesidades que se tendrán que cubrir. Asimismo, habrá que tener en cuenta la viabilidad que tendrá desarrollarlo en un lenguaje u otro en función del tiempo estimado y el hardware utilizado [20].

La POO ofrece una serie de ventajas para el mantenimiento y la actualización del software:

- Modularidad: Facilita la localización y corrección de errores.

- Encapsulación: Permite realizar cambios sin afectar otras partes del sistema.
- Herencia y composición: Fomentan la reutilización de código y la creación de jerarquías de objetos.

Estas características hacen que la POO sea la opción preferida para proyectos que requieren un alto nivel de mantenibilidad y adaptabilidad.

Otro aspecto importante para contemplar es el dominio que se tenga en los lenguajes de programación considerados para el proyecto. Para la programación se debe contemplar que se van a utilizar dos sistemas operativos diferentes por lo que se tendrá que plantear utilizar dos sistemas de programación o la versión del Instituto Nacional de Estándares Americano [21].

Se elige el lenguaje C++ por el dominio que se tiene, y además es óptimo en el manejo de memoria por su control de bajo nivel que es acceso directo a la memoria, optimización a nivel de bits, compilación a nivel de máquina, multiparadigma tanto procedural como orientado a objetos, portabilidad.

En la tarjeta de desarrollo las herramientas de programación utilizadas fueron las del sistema operativo que es el compilador C++ inherente a Linux utilizando las librerías ANSI en un ambiente de texto y en el lado de la computadora en ambiente Windows gráfico el desarrollo fue con C++ Builder utilizando la librería Indy que tiene algunos componentes que facilitan el desarrollo del software para las conexiones.

En la selección del protocolo a utilizar en la conexión, se contemplaron dos: TCP es orientado a la conexión y garantiza la entrega confiable y ordenada de datos, mientras que UDP(User Datagram Protocol) es sin conexión i.e. no hay handshake ni una garantía en la entrega de datos, priorizando la velocidad y la eficiencia con tolerancia a la pérdida ocasional de datos.

El protocolo de la conexión será TCP/IP ya que se espera no perder datos al momento de la transmisión, garantizando la veracidad de la información. Se implementará una red totalmente independiente de la establecida por los puertos ethernet o WiFi que pudiesen existir en la tarjeta de desarrollo con la finalidad de tener un cierto nivel de seguridad para no recibir ataques ya que esta conexión sería local entre la computadora y el PLC.

Preparación del entorno

Para el desarrollo del software que se utilizará en el proyecto, se divide el sistema en 4 componentes, mostrados en la figura 1.

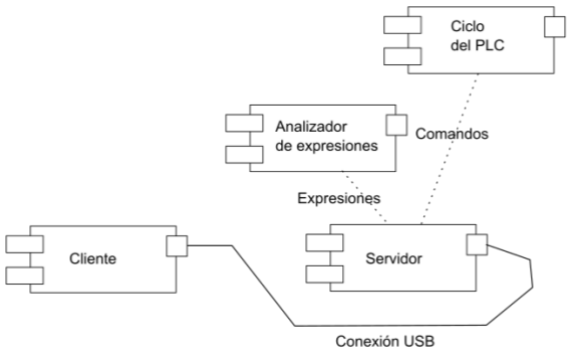

Figura 1 Componentes del sistema

Lo primero que se tiene que realizar es la instalación del sistema operativo en la Raspberry pi Zero y configurar el puerto WiFi para conectarse en la red, en el caso de este estudio se le asignó la dirección 192.168.100.21 haciendo hincapié en las direcciones IP ya que lo que se quiere es establecer dos redes, aquella conectada a internet con ethernet o WiFi y la otra local establecida con el puerto USB OTG.

Para la conexión con el puerto USB OTG, lo primero que se tiene que hacer es configurar el sistema operativo de tal manera que este sea reconocido como un puerto ethernet con una dirección totalmente diferente y nuevamente en el caso de este estudio, se asigna a la tarjeta en el ethernet del puerto USB OTG la dirección 192.168.7.3 que como se puede ver es una red diferente a la red conectada a Internet tomando en cuenta que la máscara de subred en ambas redes es 255.255.255.0. Hay que configurar la dirección de la computadora en esta conexión, fijándola en la dirección 192.168.7.1

a) Protocolo TCP para Garantizar la Veracidad:

El protocolo TCP (Transmission Control Protocol) asegura que los datos transmitidos entre el cliente y el servidor sean entregados de forma ordenada y sin

errores. Este protocolo incluye un proceso de comprobación de errores que garantiza la integridad de los datos al retransmitir cualquier paquete perdido o dañado durante la comunicación. Este mecanismo asegura que la información recibida sea fiel a la enviada, evitando inconsistencias en el intercambio de datos.

b) Mecanismos de Ajuste y Calibración:
Se implementaron procedimientos para verificar la capacidad y estabilidad de la conexión, como la reserva de memoria en el cliente antes de recibir datos y la confirmación en cada paquete recibido por el servidor. Además, el sistema utiliza métricas de desempeño, como la comparación entre la velocidad teórica máxima y la velocidad real medida durante las pruebas, para detectar desviaciones en la transmisión. Este enfoque permite ajustar parámetros como el tamaño de los paquetes (MTU) o los buffers de memoria para optimizar la conexión.

c) Pruebas de Validación:
Durante las pruebas funcionales, se verificó que los datos enviados coincidieran exactamente con los recibidos mediante controles cruzados entre el cliente y el servidor. Por ejemplo, se utilizaron archivos de tamaño conocido para asegurar que la transmisión se completara sin pérdidas ni modificaciones. Estas pruebas actúan como un proceso de calibración para confirmar que el sistema opera dentro de los parámetros esperados.

Finalmente, para probar esta conexión, se utiliza el protocolo Shell Seguro (SSH por sus siglas al inglés: Secure Shell) ya sea por WiFi con la dirección raspberrypi en donde hay que tener instalado en la computadora el servicio Bonjour que permite que los dispositivos se encuentren y comuniquen entre sí de manera automática sin necesidad de una configuración manual complicada en cuyo caso debe de existir un ruteador o switch. La otra conexión por probar es la generada con el puerto USB OTG con dirección 192.168.7.3 y el puerto USB de la computadora con dirección 192.168.7.1, la figura 2 muestra las posibles conexiones que se pueden realizar.

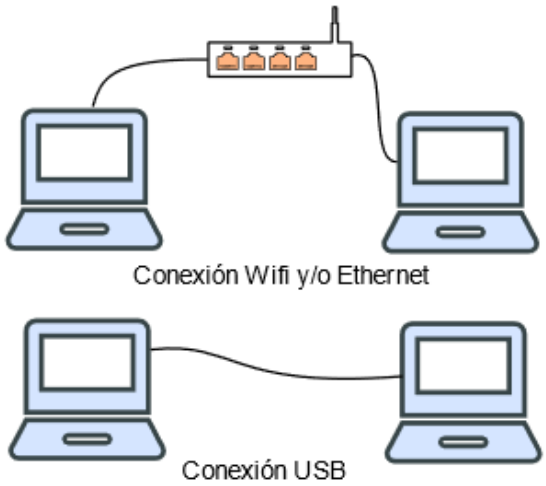

Figura 2 Conexiones PC a Raspberry Pi Zero

Diseño y desarrollo del software
En la primera fase del desarrollo, se configuró un entorno de pruebas local. Se implementó un servidor TCP en una tarjeta de desarrollo con Linux, utilizando ANSI C++ para crear las clases necesarias. Paralelamente, se desarrolló un cliente TCP en una computadora con Windows, aprovechando los componentes de la librería Indy en C++ Builder. Ambos sistemas se conectaron directamente en forma local, estableciendo una comunicación fiable a través del protocolo TCP.

Una vez desarrollados los sistemas cliente-servidor locales, se procedió a conectar la computadora con la tarjeta de desarrollo. Para ello, se adaptó el servidor de la tarjeta y el cliente de la computadora, estableciendo así una comunicación directa entre ambos dispositivos.

Para la comunicación entre el cliente y el servidor, se desarrollaron unos protocolos para el envío y recepción de datos, considerando en todo momento el protocolo TCP utilizado y entonces, antes de enviar un archivo, el servidor informa al cliente del tamaño total de los datos a transferir. Con esta información, el cliente reserva la cantidad de memoria necesaria para recibir el archivo por completo, tal como se muestran los diagramas de las figuras 3 y 4.

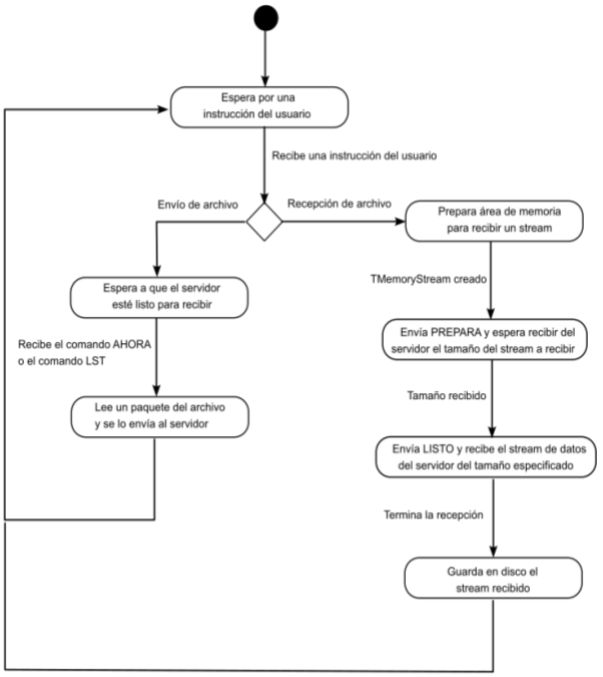

Figura 3 Diagrama de actividades E/S en el cliente

Dado que el servidor cuenta con recursos limitados (memoria) y utiliza C++ estándar, se requirió un enfoque diferente para la gestión de la conexión en comparación con el cliente (C++ Builder, Indy). Se inició un análisis detallado de cómo Indy maneja el envío de datos para poder replicar esta funcionalidad en el servidor. Debido a la naturaleza orientada a objetos de Indy, muchas operaciones se encapsulan en métodos de objetos, lo que simplifica su uso. Sin embargo, en C++ estándar, es necesario implementar manualmente la creación de objetos y sus métodos. La figura 4 muestra la implementación de la transmisión y recepción de datos en el servidor.

La figura 4 muestra una función que se ejecuta en un hilo independiente, lo que permite recibir archivos de forma concurrente sin afectar el funcionamiento general del servidor. Dentro de esta función, se implementa un ciclo que recibe paquetes de un tamaño fijo. Cada paquete recibido se almacena en el archivo y se envía una confirmación al cliente. El proceso continúa hasta que se recibe un paquete de tamaño inferior al MTU, lo que indica el final de la transferencia.

El cliente envía los datos en paquetes, esperando una confirmación del servidor antes de enviar el siguiente. Este proceso se repite en un ciclo, como se muestra en los diagramas de las figuras 3 y 4. La transferencia se detiene cuando el servidor recibe un paquete de tamaño inferior al MTU, indicando el final de los datos.

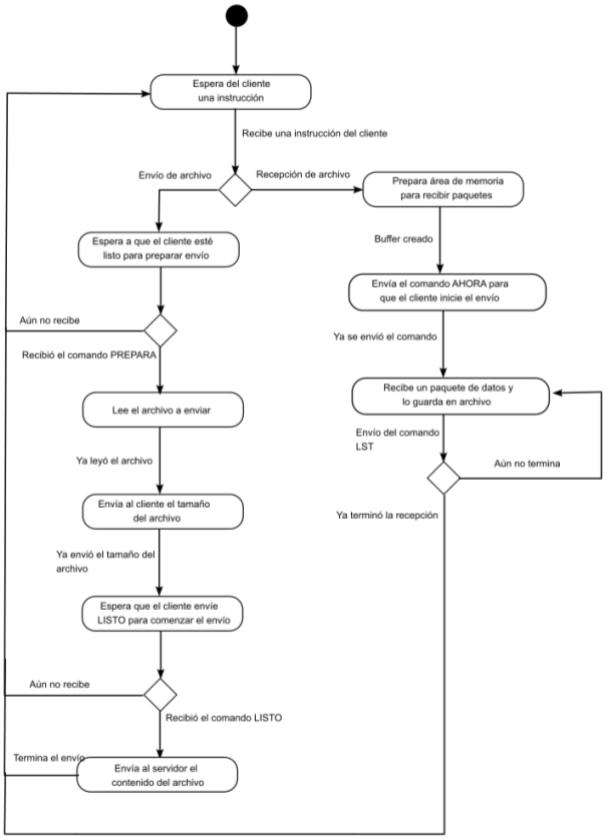

Figura 4 Diagrama de actividad E/S en el servidor

La figura 4 muestra el diagrama de actividad entrada/salida en el servidor, diseñada para enviar los datos del archivo sin esperar confirmaciones del cliente. Esta estrategia se basa en las premisas de que el cliente cuenta con los recursos necesarios (memoria y velocidad de procesamiento) para recibir los datos de manera continua, sin interrupciones y que el envío se hace a través del protocolo TCP.

Para el programa del usuario que llevará la automatización de un proceso industrial, es menester desarrollar la lógica de control del PLC, se propone una arquitectura basada en el modelo de programación IEC 61131-3. Como primer prototipo, se implementarán tres clases fundamentales: un analizador léxico, una colección de bloques de función y una clase encargada de gestionar el ciclo de escaneo. Esta estructura proporcionará una base

sólida para la creación de programas modulares y reutilizables.

El analizador se encargará de transformar las expresiones lógicas escritas en notación infija a su equivalente posfija. Esta representación posfija será evaluada posteriormente para obtener el resultado de la expresión. Los bloques de función, por su parte, representarán los elementos básicos de la expresión (operadores, variables) y estarán listos para ser ejecutados.

La expresión lógica, inicialmente expresada en notación infija, se transforma en su notación equivalente posfija. Esta representación posfija facilita la evaluación secuencial de la expresión. Cada elemento de la expresión posfija ya sea un operando o un operador, se mapea a una función unaria o binaria correspondiente. Estas funciones ejecutan la operación indicada y devuelven el resultado, permitiendo así evaluar completamente la expresión lógica.

RESULTADOS Y DISCUSIÓN

Para la evaluación del comparativo, se presentan algunas tablas que muestran el comportamiento de la transmisión de datos entre diferentes equipos. La evaluación se centra en varios factores clave: El valor máximo de transmisión, la memoria disponible y la transmisión con diferentes cables uno de 30 cm y el otro de 150 cm; además de considerar el trabajo del ciclo de escaneo durante las transmisiones para establecer un ambiente de trabajo similar al que podría tenerse en una línea de producción.

El MTU es el tamaño máximo de datos que se puede enviar en un solo paquete TCP/IP Donde se debe considerar los bytes utilizados por los protocolos TCP e IP, teniendo la oportunidad de enviar paquetes de 1400 bytes como máximo. En esta evaluación, se realizaron pruebas conectando dos computadoras usando Ethernet y WiFi 5G. Las pruebas se realizaron con paquetes de 512 bytes y 1024 bytes. La dirección de transmisión fue de cliente a servidor. Los resultados se presentan en las tablas 1 a la 9, considerando significativos los valores Mbps como resultado de la velocidad de transmisión. Ambas pruebas son en dirección cliente-servidor (CS), donde el servidor, al tener menor capacidad de memoria, presenta una mayor lentitud en la recepción continua.

Para las pruebas de MTU, se utilizó una computadora con Intel i5 con 8 GB de RAM como cliente y una tarjeta Raspberry Pi Zero 2W con 512 MB de RAM y el envío de archivos se hizo del cliente al servidor. En la tabla 1 se muestran los resultados comparativos de velocidad de transmisión por paquetes de envío con cable 30 cm.

Tabla 1 Comparativo de velocidad de transmisión por paquetes

Tamaño de archivo (bytes)	Tiempo medido (segundos)	Mbps	Sentido	MTU
8,227	0.0811	0.811	C -> S	512
12,530,642	103.4820	0.969	C -> S	512
67,559,016	560.5837	0.964	C -> S	512
8,227	0.0181	3.639	C -> S	1024
12,530,642	27.6946	3.620	C -> S	1024
67,559,016	151.4606	3.568	C -> S	1024
243,177,832	540.5464	3.599	C -> S	1024

La limitación de tamaño de paquete a 512 bytes impidió completar la transmisión de archivos mayores a 200 MB, pero cuando el paquete fue de 1024, el archivo fue enviado sin ningún problema. En el resumen de la tabla 2 se ve que la diferencia entre los paquetes de 512 y el doble de 1024, indica que utilizando el paquete de 1024 se tiene tendencia de transmisión de datos 3.7 veces mayor al de 512, indicando que, a mayor cantidad de bytes en el paquete, mejora la velocidad de transmisión.

Tabla 2 Relación entre paquetes de 512 y 1024

Tamaño de archivo	Mbps con MTU=512	Mbps con MTU=1024	Relación
8,227	0.811	3.639	4.480
12,530,642	0.969	3.620	3.7358
67,559,016	0.964	3.568	3.7012
243,177,832	0.000	3.599	

Las siguientes pruebas de memoria se realizaron con una computadora con 4 GB de RAM y un servidor con 512 MB de RAM. La prueba se realizó en dos direcciones: cliente-servidor (CS) y servidor-cliente (SC). En la dirección cliente-servidor, el servidor procesa y almacena cada 1024 bytes recibidos, mientras que, en el cliente, con más memoria, el

archivo completo se recibe y se procesa al final. Los resultados de estas pruebas se muestran en la tabla 3, en la dirección SC la velocidad es 71.46 veces mayor que en la dirección CS.

El siguiente resultado se obtiene formando una conexión con una computadora con Intel i5 con 8 GB de RAM como cliente y una Raspberry Pi zero 2W con 512 MB de RAM como servidor con un cable USB-A a micro-USB de 30 cm. Para hacer una comparación de recibir con la computadora y recibir con la tarjeta de desarrollo y fijando el paquete de envío a 1024 bytes.

Tabla 3 Comparativo de velocidad contra memoria

Tamaño de archivo (bytes)	Tiempo medido (segundos)	Mbps	Sentido
8,227	0.0282	2.333	C -> S
12,530,642	7.641	13.119	C -> S
67,559,016	43.861	12.322	C -> S
243,177,832	155.988	12.471	C -> S
8,227	0.0024	27.423	S -> C
12,530,642	0.7020	142.800	S -> C
67,559,016	3.9800	135.800	S -> C
243,177,832	14.038	138.583	S -> C

En el resumen de la tabla 4 se ve que la diferencia entre los paquetes recibidos por el servidor y aquellos recibidos por el cliente indicando una tendencia promedio de transmisión de datos 11.19 veces mayor la recepción en la PC que en la Raspberry, indicando que, a mayor de memoria y velocidad de procesamiento, mejora la velocidad de transmisión.

Tabla 4 Resumen con cable de 30 cm

Tamaño de archivo	Mbps con C ->S	Mbps con S ->C	Relación
8,227	2.333	27.423	11.754
12,530,642	13.119	142.800	10.885
67,559,016	12.322	135.800	11.021
243,177,832	12.471	138.583	11.112

El siguiente resultado se obtiene formando una conexión con una computadora con Intel i5 con 8 GB de RAM como cliente y una Raspberry Pi zero 2W con 512 MB de RAM como servidor con un cable USB-A a micro-USB de 150 cm. Para hacer una comparación de recibir con la computadora y recibir con la tarjeta de desarrollo y fijando el paquete de envío a 1024 bytes.

Tabla 5 Comparativo con memoria a 64 bits cable 150 cm

Tamaño de archivo (bytes)	Tiempo medido (segundos)	Mbps	Sentido
8,227	0.0172	3.827	C -> S
12,530,642	7.181	13.960	C -> S
67,559,016	33.787	15.996	C -> S
243,177,832	135.356	14.373	C -> S
8,227	0.0024	27.423	S -> C
12,530,642	0.5926	169.162	S -> C
67,559,016	3.7680	143.437	S -> C
243,177,832	11.9910	162.240	S -> C

En el resumen de la tabla 6 se ve que la diferencia entre los paquetes recibidos por el servidor y aquellos recibidos por el cliente indicando una tendencia promedio de transmisión de datos 9.88 veces mayor la recepción en la PC que en la Raspberry, indicando que, a mayor de memoria y velocidad de procesamiento, mejora la velocidad de transmisión; pero comparado con el ejercicio del cable de 30 cm, la transmisión del cliente al servidor y viceversa con el cable de 150 cm es más estable y la diferencia entre ambos sentidos es más pareja.

Tabla 6 Resumen con cable de 150 cm

Tamaño de archivo	Mbps con C ->S	Mbps con S ->C	Relación
8,227	3.827	27.423	7.166
12,530,642	13.960	169.162	12.118
67,559,016	15.996	143.437	8.96
243,177,832	14.373	162.240	11.287

Comparando los resultados de un cable de 30 cm contra otro de 150 cm, el promedio de velocidad en el sentido cliente-servidor fue de 10.061 Mbps para el cable de 30 cm y de 12.03 para el de 150 cm; en el sentido servidor-cliente fueron velocidades de 111.15 Mbps y 125.57 Mbps respectivamente; en la tabla 7 se muestra el comparativo con respecto a las velocidades ideales establecidas por los fabricantes de 480 Mbps para el USB versión 2.0, 1000 Mbps para ethernet con cable categoría 6 y 10 Mbps para la tecnología RS-485, estableciendo la eficiencia en velocidad sin pérdida de datos entre las transmisiones utilizando cables de 30 cm y 150 cm.

Tabla 7 Eficiencia en las transmisiones USB-OTG

Transmisión	Mbps	USB 2.0 480 Mbps	Ethernet cat. 6 ó WiFi 1000 Mbps	RS485 10 Mbps
S->C Cable 30 cm	111.15	23.16 %	0.111 %	1111%
S->C Cable 150 cm	125.57	26.16 %	0.126 %	1256%
C->S Cable 30 cm	10.06	2.09 %	0.010 %	101%
C->S Cable 150 cm	12.03	2.51 %	0.012 %	120%

Estos resultados nos indican estar en un punto medio entre dos tecnologías como lo son ethernet, WiFi y RS-485; recalcando la importancia del medio físico de transmisión, ya que a pesar de que el cable de 150 cm es más largo, arrojó velocidades mayores; las pruebas también demostraron que la capacidad de memoria tiene un efecto considerable en el rendimiento de la transmisión. El cliente con mayor cantidad de memoria RAM mostró una mayor velocidad en comparación con el servidor que tiene menos RAM. Cuanta más RAM tenga la PC, más datos podrá almacenar temporalmente, así como menos interrupciones entre los envíos de paquetes, se resalta la importancia de una memoria adecuada para manejar transmisiones continuas de datos.

CONCLUSIONES

Los resultados obtenidos confirman que la tecnología USB-OTG es una solución efectiva y segura para mejorar la comunicación en sistemas industriales que utilizan PLCs. Aunque su velocidad de transmisión es menor en comparación con otras tecnologías como Ethernet o Wi-Fi, la optimización del tamaño de paquetes (MTU de 1024 bytes) y la implementación del protocolo TCP permiten garantizar la integridad de los datos y una operación confiable, incluso en escenarios industriales críticos.

El estudio destaca la influencia de factores clave como la memoria RAM y la capacidad de procesamiento en el rendimiento de la transmisión. Los dispositivos con mayor capacidad lograron velocidades significativamente superiores, evidenciando la importancia de considerar el diseño del hardware en la configuración de sistemas que requieran altas demandas de comunicación. Asimismo, la calidad del medio físico, como el tipo y longitud de los cables, tiene un impacto directo en la estabilidad y eficiencia de la transmisión.

Finalmente, la implementación de USB-OTG como alternativa de comunicación privada y directa refuerza su relevancia en la Industria 4.0. Su capacidad para operar de manera aislada de redes externas reduce significativamente el riesgo de ciberataques, posicionándola como una opción práctica y escalable para la integración de sistemas de automatización industrial.

REFERENCIAS

Las referencias fueron verificadas en Trustpilot, linkedin, Google y Bing, para asegurar la veracidad de la información y sus fuentes.

1. Innova Más, (2023), CONTROLADOR LÓGICO PROGRAMABLE (PLC), https://innovamas.nakasawaresources.com/controlador-logico-programable-plc/ (accessed Sep. 21, 2024).
2. Namekar and R. Yadav (2020), Programmable Logic Controller (PLC) and its applications, S. A., International Journal of Innovative Research in Technology (IJIRT), vol. 6, no. 11, pp. 372-376, 2020.
3. IONOS Digital Guide, (2023), ¿Qué significa USB? Descubre qué es Universal Serial Bus, https://www.ionos.mx/digitalguide/servidores/know-how/que-es-usb/#:~:text=USB%2C%20cuyas%20siglas%20significan%20%E2%80%9CUniversal (accessed Sep. 22, 2024).
4. Pardo D., (2021), Tipos de USB: la lista definitiva para conocerlos todos, Pandora FMS, The Monitoring Blog, https://pandorafms.com/blog/es/tipos-de-usb/ (accessed Sep. 23, 2024).
5. Burn I., Cómo conectar dos PC: por USB, Wi-Fi, cable de red, entre sí, Ccm.net, (2014). https://es.ccm.net/ordenadores/redes/1300-como-conectar-dos-pc-con-un-cable-usb/#google_vignette (accessed Sep. 23, 2024).
6. Enrique, J. N. (2021). Módulo Terminal Remoto, para la adquisición de datos, monitoreo y control de procesos Agroindustriales. AgriculTIC. Ingeniare. Revista Chilena de Ingeniería, 245-264. doi: https://doi.org/10.4067/s0718-33052021000200245
7. Fernández, A. (2018). Remote supervision and control based on wireless technology to operation of central pivot irrigation machine. S istemas y Telemática, 16(44), 63-74.
8. R. K. K, R. M. (2021). Embedded Real Time Media over Ethernet via USB-OTG. IEEE Mysore Sub Section International Conference (MysuruCon), 734-737.
9. Cameron, N. (2021). USB OTG communication. In: Electronics Projects with the ESP8266 and ESP32. Apress. doi: https://doi.org/10.1007/978-1-4842-6336-5_13

10. Fernández Y., (2021), Tipos de USB: estándares, conectores y características de cada uno, Xataka, https://www.xataka.com/basics/tipos-usb-estandares-conectores-caracteristicas-cada-uno
11. USB Gadget API for Linux — The Linux Kernel para asegurardocumentation. (2024). Kernel.org. https://www.kernel.org/doc/html/v5.1/driver-api/usb/gadget.html
12. Raspberry Pi. (2023). Raspberry Pi Documentation - Raspberry Pi OS. Www.raspberrypi.com. https://www.raspberrypi.com/documentation/computers/os.html
13. Abdelsalam, M. (2024). How PLC communication affects cycle load, Siemens PLC, CPU communication load, TIA portal, PCS7, PLC overload, PLC hardware configuration, S7-1500, S7-1200. Linkedin.com. https://www.linkedin.com/pulse/how-plc-communication-affects-cycle-load-mohamed-abdelsalam-tgvvf/
14. Chuadhry Mujeeb Ahmed, Martin Ochoa, Jianying Zhou, and Aditya Mathur. (2021). Scanning the Cycle: Timing-based Authentication on PLCs. In Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (ASIA CCS '21). Association for Computing Machinery, New York, NY, USA, 886–900. https://doi.org/10.1145/3433210.3453102
15. Liang Jiang ping., (2017), Program design of PLC scanning cycle analysis method [J]. China Southern Agricultural Machinery, 48 (13): 118+121.
16. Duan, W. (2020). Research on Programming Failure of PLC Direct Translation Method. Academic Journal of Engineering and Technology Science, 3(5), 13-20.
17. L. Da Xu, W. He and S. Li, (2014) "Internet of Things in industries: A survey", IEEE Trans. Ind. Informat., vol. 10, no. 4, pp. 2233-2243.
18. L. Li, S. Li and S. Zhao, (2014), "QoS-aware scheduling of services-oriented Internet of Things", IEEE Trans. Ind. Informat., vol. 10, no. 2, pp. 1497-1505.
19. Pastor, J. (septiembre 24, 2022). La luz está tan cara que averiguar cuál es el lenguaje de programación más eficiente es una buena idea. Xataka. https://www.xataka.com/aplicaciones/luz-esta-cara-que-averiguar-cual-lenguaje-programacion-eficiente-buena-idea
20. Eficiencia de los lenguajes de programación (2022). Undefined World. Undefined World. https://www.undefinedworld.com/blog/eficiencia-lenguajes-programacion
21. What is C++ & How It Compares to Other C Programming Languages | Simplilearn. (2022, February 16). Simplilearn.com. https://www.simplilearn.com/what-is-cpp-programming-article