$DIFU_{100}ci@$
Revista de Difusión Científica, Ingeniería y Tecnologías
Vol. 15, No. 3, Septiembre - Diciembre 2021
ISSN:2007-3585
Artículo arbitrado
Publicado: 31 de diciembre de 2021

System Development with PLC Logo 8 Siemens, Raspberry Pi4 and ESP8266-12E

Desarrollo de Sistema con PLC Logo 8 Siemens, Raspberry Pi4 y ESP8266-12E

Jaime Jalomo Cuevas[1], **Rubén Reyes Rubio**[1], **Ramiro Rodríguez Pérez**[1], **Leopoldo Esesarte Rodríguez**[1], and **Juan Francisco Palomino Bernal**[*1]

[1] *Tecnológico Nacional de México/Instituto Tecnológico de Ciudad Guzmán, Departamento de Eléctrica y Electrónica,*
Av. Tecnológico 100, Carretera al Fresnito, Ciudad Guzmán, Jal., México, 49100.
jaimejc@itcg.edu.mx, {ruben.rr,ramiro.rp,leopoldo.er}@cdguzman.tecnm.mx,jufrapabe10@gmail.com

Abstract

The present study was developed using an experimental and comparative methodology for the development of a communication system based on both wired and wireless technology, through the integration of different technologies or devices for industrial use such as a Siemens Logo 8 PLC, up to free access as the Raspberry PI 4 and the NodeMCU ESP8266-12E Board and the Node-Red and Mosquitto (MQTT) software, managing to develop a 4-phase communication between the devices, resulting in a reliable communication system, in real time and data processing of Industrial grade, for monitoring and control of any type of variable that can be processed by both the PLC and the NodeMCU ESP8266-12E development board.

Keywords— Open Source, Comunication, Integration, Mosquitto, Node-Red.

Resumen

El presente estudio se desarrolló utilizando una metodología experimental y comparativa para el desarrollo de un sistema para comunicación basado en tecnología tanto alámbrica como inalámbrica, mediante la integración de diferentes dispositivos de uso industrial como un PLC Siemens Logo 8, hasta de acceso libre como la Raspberry PI 4 y la Placa NodeMCU ESP8266-12E y las herramientas de software Node-Red y Mosquitto (MQTT) logrando desarrollar una comunicación de 4 fases entre los dispositivos, resultando un sistema de comunicación confiable, en tiempo real y proceso de datos de grado industrial, para monitoreo y control de cualquier tipo de variable que puedan procesar tanto al PLC, como a la placa de desarrollo NodeMCU ESP8266-12E.

Palabras clave— Fuente Libre, Comunicación, Integración, Mosquitto, Node-Red.

I. Introducción

Actualmente, existen numerosas investigaciones referentes a integración de tecnologías para monitoreo, control y conexión de dispositivos. No obstante, algunos sistemas quedan limitados desde la calidad en dispositivos para generar un control confiable de grado industrial y la posible pérdida de información al ser transmitida mediante protocolos de comunicación estándar, en el presente trabajo se utilizan diferentes dispositivos para desarrollar un sistema completo independientemente la variable a controlar, el presente estudio abarca la integración de diferentes dispositivos electrónicos. El uso y desarrollo se limita a las posibilidades de la placa de desarrollo ESP8266-12E y el PLC Logo 8 de Siemens, de esta manera, el uso del PLC, aplicará como un dispositivo de grado industrial para procesamiento, control y envío de datos, continuando con una transmisión de datos por medio de software para su conexión (Node-Red y Mosquitto) y envío datos en tiempo real, en un Red Local, impactando en la reducción de manera significativa el coste del sistema desarrollado, usando una conexión Ethernet, evitando el uso del servidor externo (MQTT adafruit.io cloud) y ejecutando instrucciones diseñadas

[*]Autor de correspondencia

previamente, basadas en los datos recibidos por la placa ESP8266-12E.

El PLC Logo 8, desempeñará cualquier tipo de control deseado y desarrollará el papel en el presente trabajo como esclavo y será mediante comunicación TCP/IP, con un módem y protocolo Modbus RS-232, como Anwaruddin [1], lo realizó de manera alámbrica. Posteriormente, para la primera fase de comunicación, se utiliza el software Node-Red [2], mediante el puerto 512 de ethernet para transmisión de datos de manera local como lo realizó A. I. Marosan [3], pero sin utilizar un Arduino Mega, posteriormente, se convierten, procesan y envían a leguaje MQTT, como lo elaboró Laluma R.H. [4], a diferencia con el presente, no se usará el servicio adafruit.io cloud, reduciendo el coste del uso de la cuenta, también se usará el puerto TCP/IP 1883 del Broker Mosquitto, para establecer como publicador al PLC mediante Node-Red, transmitiendo la información al mismo, para después ser enviados y recibidos mediante Wifi con el chip ESP8266-12E, de la misma manera que lo desarrolló Gupta en 2020 [5], a diferencia que se usará el puerto TCP/IP 1883, para escuchar mensajes de publicadores y subscriptores [6], logrando el uso del Broker sin el coste de una cuenta adafruit.io cloud, concluyendo con la conexión como subscriptor en el broker el ESP8266-12E.

II. Herramientas de software y hardware

II.1. PLC Siemens Logo 8

El PLC Logo 8 de Siemens por sus siglas en inglés (Programable Logic Controleer), es un dispositivo de gama industrial, principalmente fabricado por la empresa Siemens para proyectos de domótica (automatización de hogares). El Logo 8 cuenta con una pantalla interfaz empotrada en el dispositivo, completas opciones de comunicación a través de Ethernet, un servidor web integrado además de salidas adicionales en los módulos digitales y analógicos. La pantalla contiene 6 líneas con 16 caracteres por línea. También la interfaz Ethernet y el servidor web se encuentran integrados en el PLC, resultando en una utilidad alta en esta clase de dispositivos. Además de poder ejecutarse bajo versiones de 32 y 64 bits de Windows, incluyendo Windows 10, así como Mac y Linux. También cuenta con servidor web, permitiendo la supervisión y el control del módulo lógico en remoto a través de internet. La creación de páginas web no requiere conocimientos especiales de programación existiendo la posibilidad de elegir libremente sus opciones preferidas de visualización. Importante mencionar, la Comunicación integrada Modbus TCP/IP y sincronización de fecha y hora a través de NTP (Network Time Protocol). Siendo el PLC más accessible en relación asequible/calidad.

II.2. Node-Red

Node Red es una plataforma de desarrollo mediante el uso en Node.JS y conjuntamente funciona como un motor de ejecución asíncrono de Javascript para servidores. Node-Red es Open-Source y fue diseñado por la empresa IBM hace unos años para simplificar el desarrollo de aplicaciones orientadas al manejo de eventos asíncronos, como los son las aplicaciones de IoT. Su uso puede ir desde una aplicación simple en el hogar hasta plataformas IoT en la nube. La interfaz web corre por defecto en el puerto 1880. El software cuenta con bloques de entrada, de salida y otros que procesan información o ejecutan diversas acciones y cada bloque puede ejecutar cualquier código javascript que se desee.

Con la interconexión de los bloques, se hace que los eventos y los datos generados en cada uno, interactúe con los otros. De esta manera se forma un “Flow” o flujo de información. Mediante uno o varios flows se puede desarrollar una aplicación IoT que reciba, procese, transmita y presente información.

II.3. Mosquitto

La arquitectura de Mosquitto o MQTT (Message Queue Telemetry Transport), sigue una topología de estrella, con un nodo central que hace de servidor o “broker” normalmente con una capacidad teórica de 10000 clientes. El “broker” es el encargado de gestionar la red y de transmitir los mensajes, para mantener activo el canal, los clientes mandan periódicamente un paquete de datos.

La comunicación se basa en topics (tópicos) y para que un cliente tenga acceso a la información, debe estar subscrito al tópico sin importar cuantos clientes estén siguiendo el tema.

Un cliente (cualquiera) puede publicar mensajes y los nodos, que deseen recibirlo, deben estar subscrito a él. La comunicación puede ser de uno a uno, o de uno a muchos. Un tópico se representa mediante una línea de código y tiene una estructura jerárquica separada con ’/’. Ejemplificando: “sonos/sensor1/temperatura” o “sonos/sensor2/ruido”.

De esta forma se pueden crear jerarquías de clientes que publican y reciben datos.

Mosquitto, de esta manera, cuando los enfoques informáticos clásicos (servidores web, socket de red, etc.) resultan ser demasiado “pesados” por la cantidad de recursos necesarios para sostenerlos o resultan ser soluciones exageradas para resolver una simple comunicación de algunos Bytes, el protocolo MQTT resulta la mejor solución para ello.

II.4. ESP 8266-12E

El ESP 8266 es un microcontrolador que en relación costo-funcionalidad, resulta ser atractivo y asequible. Cuenta con una CPU RISC de 32 bits fabricada por Tensilica dentro del modelo Xtensa LX106, el cual trabaja a 80 MHz. También cuenta con la capacidad de soportar overclock, tanto en su CPU como en su memoria flash, aumentando se capacidad hasta los 160 MHz en su procesador y entre los 40 y 80 MHz en su memoria flash. Además, tiene una Caché de instrucciones de 64 KB y una de datos de 96 KB, capacidad externa de memoria QSPI que puede llegar hasta los 16 MB, soporte para IEEE 802.11 b/g/n, TR switch, balun, LNA, soporte para WEP y WPA/WPA2.

Los métodos de entrada se reducen a una conexión de 16 pines GPIO, otorgándole mayor flexibilidad. Además, tiene soporte para UART, SPI e I2C, donde trabaja a un voltaje entre 3V y 3,6V con una intensidad de 80mA. Para su configuración, hay numerosas plataformas, siendo la más común Arduino IDE, mediante paqueterías en específico dependiendo el proyecto desarrollado.

III. Desarrollo del sistema

Para el presente trabajo se definen como esclavo el PLC Logo 8 de Siemens y el maestro el ESP8266-12E, para realizar la conexión se desprenden 4 fases que son PLC a Node-Red de Node-Red a Mosquitto y de Mosquitto a la placa de desarrollo ESP8266-12E, en la Fig. 1, se puede apreciar el diagrama general del sistema desarrollado.

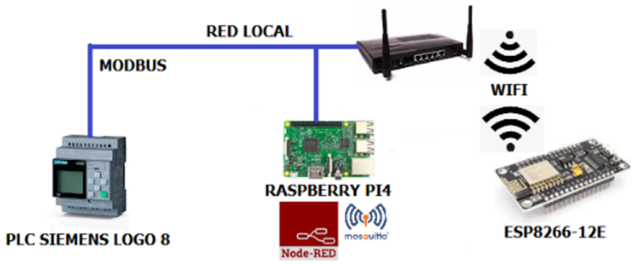

Figura 1: *Diagrama de desarrollo del sistema*

III.1. PLC Siemens Logo 8

Inicialmente, se desarrolla un diagrama de red para nuestro PLC en la interfaz del software llamado Logo SoftComfort v.8 para control y proceso de cualquier variable. Continuando la comunicación, se realiza mediante protocolo TCP/IP la correspondiente asignación de una dirección IP a nuestro PLC, mediante la interfaz del software previamente descargado e instalado. Posteriormente, se crea el servidor mediante la elaboración de un digrama de red y la configuración de conexión modbus del PLC Siemens. Importante señalar, la puerta de enlace predefinida del PLC es 255, y el puerto de conexión en el PLC Logo que corresponde al registro del puerto es 513, establecido de fábrica, debido a que el conteo de puertos inicia en 1 y no en 0, existiendo un desfase de fábrica en el conteo de puertos para comunicación Modbus. Como se puede apreciar en la Fig. 2. Posteriormente se aceptan todos los requerimientos que solicite el software, evitando algún conflicto en específico para la comunicación buscada, quedando de este modo establecida de manera satisfactoria la comunicación Modbus en el PLC Logo, para envío de datos a una red local hacia el modem, para su posterior manipulación.

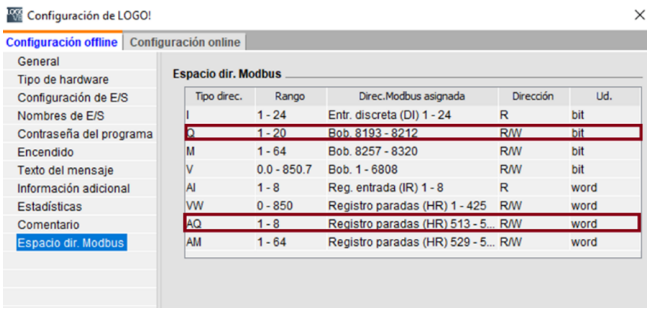

Figura 2: *Direcciones para salidas preestablecidas en comunicación Modbus*

III.2. Configuración Node-Red

Posteriormente cargado el diagrama de red con las funciones o control que se necesiten en el PLC Logo 8, se procede a establecer los parámetros para envío de datos al software Node-Red, (software precargado de fábrica en la placa Raspberry Pi4), se abre la interfaz y se utilizan los flujos mostrados en la Fig. 3, para realizar comunicación PLC/Node-Red, Node-Red/Mosquitto. Importante realizar la descarga e instalación de librerías para el uso de los flujos correspondientes al PLC Logo 8 en su versión para Node-Red, una vez descargados, se realiza la configuración de los nodos necesarios con sus puertos y direcciones correspondientes al PLC Logo 8 (desarrollada de manera detallada en la sección III.5, del presente). Para lograr establecer la comunicación, en el presente trabajo, se utilizaron 4 nodos, generando comunicación entre software y dispositivo, siendo estos: nodo Read Modbus Logo, nodo function, nodo Debug y nodo MQTT out, el primer nodo, despeña la función de leer los datos directamente del PLC Logo 8, el bloque function, desarrolla un código de decodificación para los datos en comunicación modbus, para posteriormente el nodo MQTT out envía los mismos al "broker" del software Mosquitto, (apreciable más a detalle en la sección III.5), este último, se encargará de enviar los datos hacia

el software Mosquitto, como publicador y por último el nodo debug, su función consta en ayudar al desarrollador a comprender lo que sucede en un flujo. No menos importante mencionar, el uso del nodo debug debido a sus posibilidades de depurar, solo en los puntos en los que haya agregado un nodo al flujo.

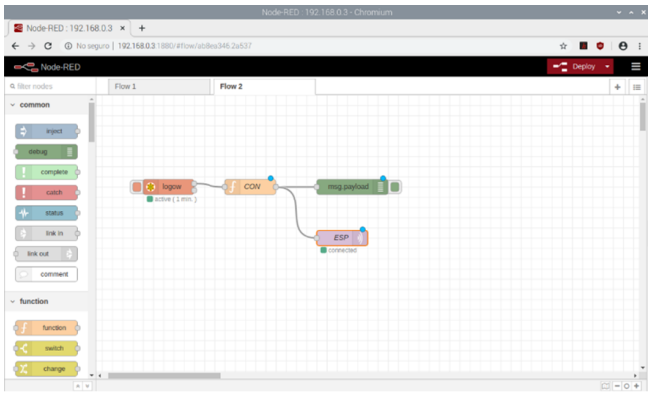

Figura 3: *Flujos y sus conexiones para envío PLC a Node-Red, Node-Red-Mosquitto*

III.3. Comunicación Mosquitto (MQTT)

Continuando, se instala el software Mosquitto y la paquetería "Clients" en la Rasberry PI4, necesario realizar la descarga del software correspondiente, de la red y posteriormente, se habilita el puerto 1883 en la configuración del software, ingresando al código fuente del software, mediante la carpeta de ".config" y declarando al puerto 1883 como "habilitado para escuchar" (desarrollado a detalle en la sección III.5), resultando en el acceso para publicadores y subscriptores en red local como se aprecia en la Fig. 4.

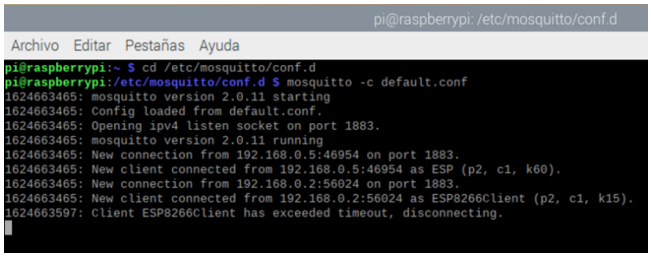

Figura 4: *Inicialización de puerto 1883 y broker Mosquitto*

III.4. ESP 8266-12E Subscriptor Mosquitto

Para recibir como subscriptor al ESP8266-12E en el entorno de Mosquitto, mediante el IDE de Arduino, se cargan las librerías correspondientes a la placa de desarrollo y se realiza el sketch correspondiente, utilizando los datos enviados desde el PLC, para su manipulación correspondiente, lo anterior dependiendo las necesidades de la variable a controlar, observándose en la Fig. 5, un ejemplo.

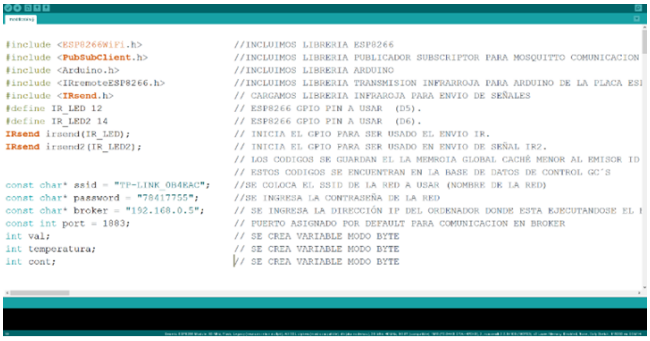

Figura 5: *Librerías incluidas en sketch para subscripción en Mosquitto del ESP8266-12E*

Cargado el sketch necesario en la placa ESP8266-12E, se alimenta la misma, mediante uno de los puertos USB de la Raspberry Pi y se inicializan las herramientas de software en la misma, resultando la conexión de 4 fases, entre los dispositivos electrónicos para el control de una variable en específico, por ejemplo: la temperatura.

III.5. Detallado particular para la configuración del sistema

Respecto al PLC Logo 8, se requiere configurar la dirección IP del mismo previamente y asignarle la salida de red como se puede visualizar en la Fig. 6, todo mediante la opción proyecto de Red, se selecciona el tipo de PLC y el tipo de comunicación denominada como servidor en comunicación modbus, habilitando el puerto y apareciendo en color amarillo un recuadro, apreciable en la Fig. 6.

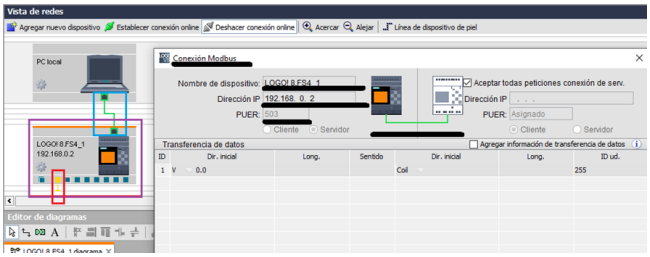

Figura 6: *Configuración parámetros PLC Logo 8 y direcciones*

En Node-Red se deben realizar las configuraciones de cada nodo a utilizar, se selecciona el nodo previamente descargado de la librería para el PLC Logo 8 llamado "Read Modbus" y se configuran los puertos y enlaces correspondientes como se muestra en la Fig. 7, asignando desde el nombre del nodo, hasta el puerto de enlace predeterminado "Unit-ld 255", una vez asignados los valores ilustrados en la Fig. 7. Seleccionamos la opción "DONE",

completando la configuración del nodo del PLC Logo 8. Se debe seleccionar obligatoriamente “Read Holding Registers” en la opción “FC”, indicando de esta manera la lectura de registro mantenido en el PLC, siendo este ultimo, el registro donde se encontrará la información necesaria desprendida del PLC, para transmitirla por el Sistema de 4 fases. Despues se configura el nodo “function”, necesario para obtener la extracción de los datos leídos del PLC Logo 8.

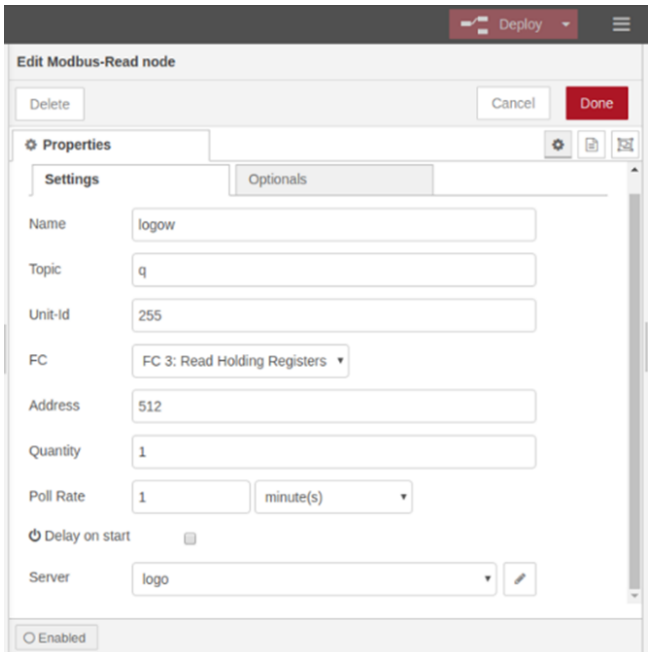

Figura 7: *Configuración nodo Modbus Read PLC Logo 8 en Node-Red*

El código mostrado en la Fig. 9, representa la función de extracción de los datos transmitidos vía Modbus hacia Node-Red para ser transmitidos posteriormente, hacia el broker de Mosquitto “MQTT”, en el código, se puede apreciar la función utlizada, necesaria, para extraer la información leída del PLC Logo 8, lo anterior, mediante la creación de una variable y asignándole el valor convertido. Para complementar el proceso correspondiente, se utiliza también el nodo “Debug”, para realizar la función de depurar y poder observar el estado del bloque “function”, en cada transmisión leída y enviada mediante el nodo descrito, para utilizarlo, solo se selecciona, se arrastra hacia la interfaz de diagrama y se integra en el diagrama de flujos; apreciable gráficamente en la Fig. 3. Respecto a la configuración, se selecciona, la categoría de “msg.payload”, en la configuración de la comunicación Modbus, sin embargo, para la presente comunicación solo es necesario uno. La configuración descrita anteriormente puede apreciarse en la Fig. 8.

Después, al nodo MQTT out (mostrado en la Fig. 3), primero se le asigna en la configuración, la dirección IP

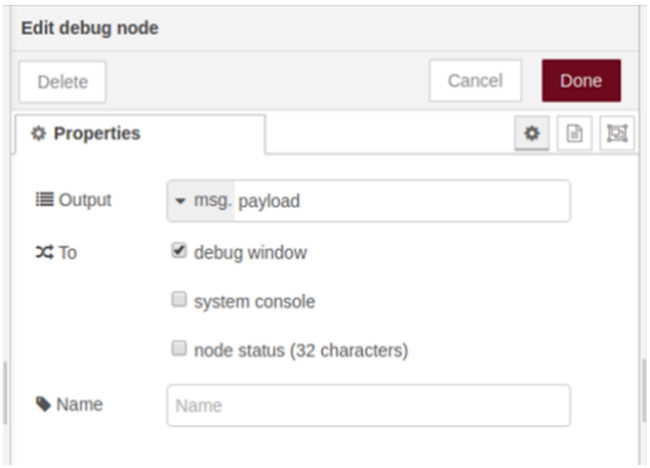

Figura 8: *Configuración nodo “Debug” en Node-Red*

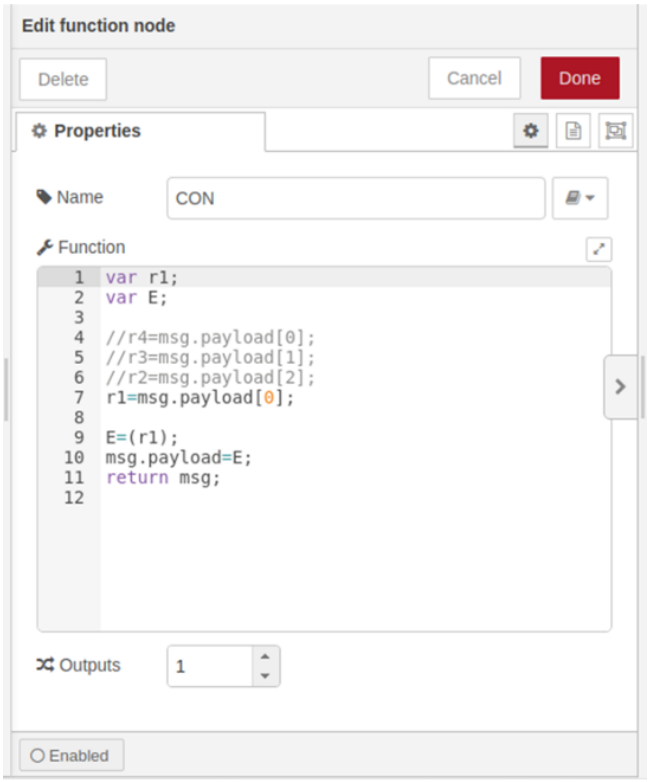

Figura 9: *Configuración nodo “Function” en Node-Red*

del dispositivo Raspberry Pi 4, para envío de datos hacia el broker de Mosquitto, después, se asigna el nombre del servidor/cliente y por último, el puerto de comunicación 1883 para “escuchar” mensajes como publicador el software Node-Red. La configuración anterior, puede apreciarse en la Fig. 10, finalizando la misma en la tecla “DONE”.

La configuración del software Mosquitto, se realiza en el dispositivo Raspberry Pi 4, mediante las líneas de

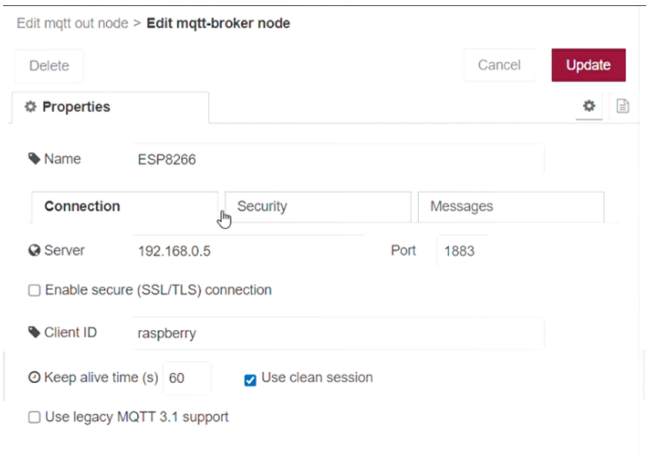

Figura 10: *Configuración nodo "MQTT out" en Node-Red*

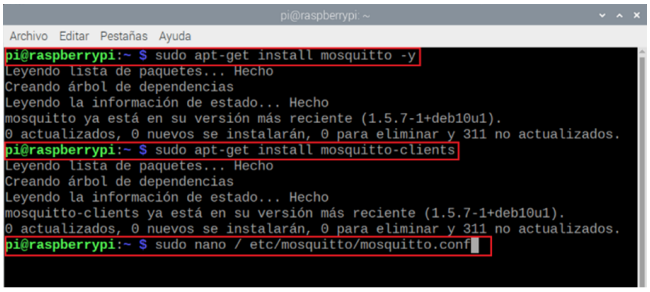

Figura 11: *Configuración software Mosquitto en Raspberry Pi 4*

Al ingresar a la configuración del código fuente del software Mosquitto, resulta necesario, depurar las líneas de código que aparecen en la misma, debido a la restricción que generan para el uso del puerto 1883; para recibir y enviar datos en una red local. Se depuran definitivamente las líneas de código descritas anteriormente, para introducir nuevas líneas de codificación, es posible observar el código mencionado en la Fig. 12, se guardan los cambios realizados y se reinicia el dispositivo Raspberry Pi4. Mediante la anterior acción, se habilita de esta manera el puerto 1883, para recibir y enviar datos mediante una red local, siendo posible utilizar el software de manera libre, sin necesitar una cuenta "adafruit.io cloud". Aceptando ahora el "broker" de Mosquitto, tanto subscriptores como publicadores.

Mediante el IDE de Arduino, se realiza la configuración para recibir datos desde el broker de mosquito vía WiFi,

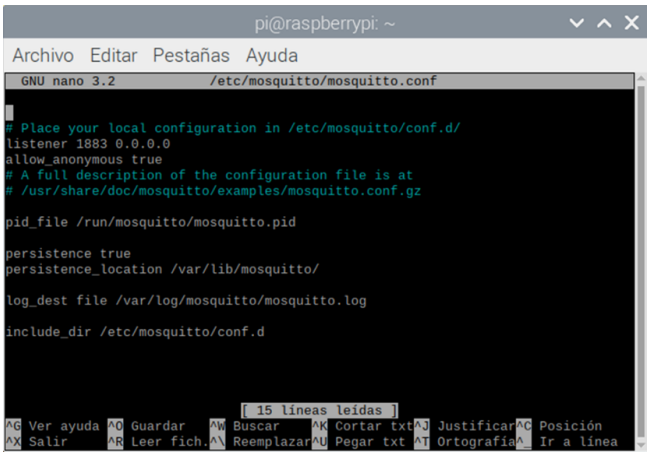

Figura 12: *Habilitación del puerto 1883 del software Mosquitto en Raspberry Pi 4*

la codificación necesaria, para establecer como subscriptor en el broker de Mosquitto al dispositivo ESP8266-12E, se observa en la Fig. 13, definiendo al dispositivo como subscriptor mediante la librería "PubSubClient", las constantes char e int, funcionan para establecer el nombre del modem, contraseña, dirección IP y el puerto de comunicación hacia el broker de Mosquitto. Después, una sección de vaciado llamada "void callback", tiene como función el solicitar envíos de datos cada segundo, continuando con la codificación "Serial.print", ejecuta la impresión de mensajes en el monitor serie y finalizando la configuración con un bucle para el procesamiento de los datos, como un string ó vector extrayéndolos desde la comunicación modbus para su posterior manipulación a la necesidad requerida del sketch.

Figura 13: *Codificación en IDE de Arduino para conexión a "Broker"*

IV. Resultados

IV.1. Interfaz Node-Red

Al realizar la conexión de los dispositivos a su alimentación correspondiente e inicializar en la Raspberry Pi

4 y las herramientas de software para ejecutar la comunicación de 4 fases, resulta una comunicación exitosa, tanto en envío de datos en tiempo real, como la recepción de los mismos. Node-Red, en su interfaz, cuenta con la facilidad de manera ilustrativa para comprobar la comunicación, entre los flujos utilizados mediante un "punto azul", confirmando de esta manera la comunicación entre PLC y el software Mosquitto, como se aprecia en la Fig. 14. Indicando el flujo "PLC Logo 8 SIEMENS", la conexión y comunicación entre el dispositivo y el software Node-Red (alojado en la Raspberry Pi4), conectándose al flujo "function" y enviando los datos codificados al flujo "MQTT" denominado en la configuración "ESP8266-12E", estableciendo la comunicación Node-Red de una manera exitosa, como un publicador para el "broker¨ de Mosquitto finalizando la comunicación por parte del software Node-Red.

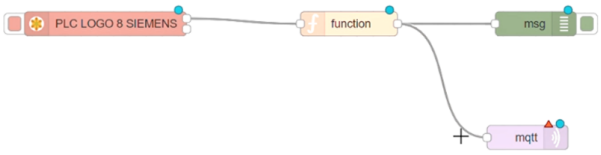

Figura 14: *Confirmación de conexión en Node-Red*

IV.2. Interfaz Mosquitto

Concerniente al software Mosquitto y a la placa de desarrollo ESP8266-12E, se puede comprobar la comunicación exitosa mediante la interfaz en el sistema Linux, como se puede observar en la Fig. 15, arrojando el software la inicialización del mismo y su versión, la lectura de la configuración realizada previamente, incluyendo la disponibilidad del Puerto TCP/IP 1883, para ser utilizado por subscriptores hacia el broker de Mosquitto.

En la Fig. 4, es posible apreciar también, mediante el último comando, la conexión de nuevos subscriptores hacia el "broker" del software (ESP8266-12E), para posteriormente y a partir de la configuración previamente realizada, emitir los envíos de datos, para su correspondiente procesamiento; a necesidad del desarrollador y sus dispositivos a utilizar.

IV.3. ESP 8266-12E con monitor serial Arduino IDE

Respecto al monitor serial de la interfaz de Arduino IDE, es posible observar el estado de la placa ESP8266-12E, como se muestra en la Fig. 16, primero ilustra el inicio con la conexión a la red WiFi, de manera exitosa,

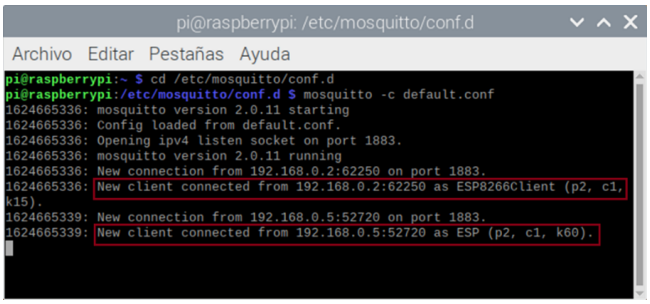

Figura 15: *Comunicación y conexiones de dispositivos en interfaz Linux de software Mosquitto*

después, la conexión al software Mosquitto, para posteriormente realizar la conexión al "broker" del software y el estado de la placa de desarrollo, al "broker" como subscriptor, para poder recibir los datos provenientes desde el PLC Logo 8, mediante la comunicación de 4 fases. Los mensajes anteriormente descritos, dependen de programar la impresión de los mismos, en el código fuente del sketch realizado, existiendo una amplia gama de opciones, para declarar, las circunstancias en que se solicite, recibir información en el monitor serial de Arduino IDE.

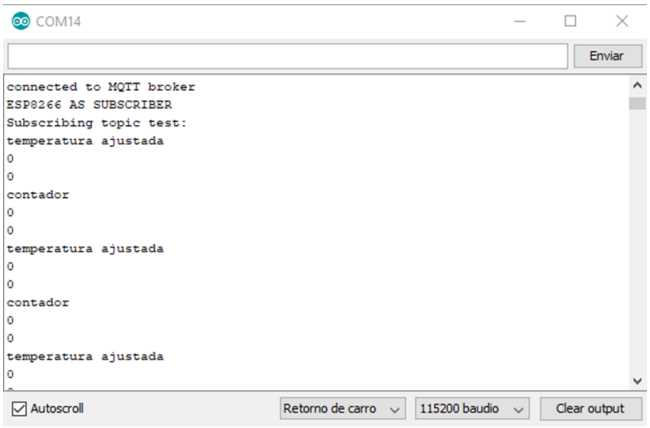

Figura 16: *Monitor serial de Arduino IDE*

V. Discusión

Existen numerosos sistemas integrados en la actualidad, el presente trabajo, se enfoca en un sistema dedicado a la integración de diferentes tecnologías, tanto de propósito industrial, como de acceso libre, produciendo confiabilidad al usar un dispositivo de gama industrial, como el PLC Logo 8, la comunicación en tiempo real al utilizar las herramientas de software de acceso libre como Node-Red, Mosquitto y la placa de desarrollo NodeMCU ESP8266-12E. Existen diferentes herramientas de software para establecer la comunicación entre dispositivos en el presente, el uso del software Mosquitto, resulta el más

conveniente para asegurar, tanto la recepción de información por parte de publicadores, como la recepción de la misma hacia subscriptores, comprobado por Hervas C. [7], en 2018, mediante un análisis comparativo, comprobando la comunicación confiable del software Mosquitto, sin pérdida de datos por parte del software.

V.1. Comprobación Node-RED

La interfaz de Node-Red, proporciona la identificación del funcionamiento de los flujos correspondientes, apreciable en la Fig. 16, desde ventanas emergentes, hasta indicadores al perímetro de cada nodo; proporcionando certeza desde la configuración de flujos, hasta el éxito en las conexiones diseñadas en la interfaz, en la Fig. 17, es posible apreciar, desde mensajes de error en los flujos, como "waiting" y "disconected", en el flujo "read PLC Logo 8" y el flujo "temperatura de MQTT out" y también, una ventana confirmando el error en la configuración de nodos, es posible realizar un análisis comparativo de la Fig. 14 y la Fig. 17, confirmando, el correcto funcionamiento del sistema en la Fig. 14, existiendo comunicación desde en PLC Logo 8, Node-Red y el envío de datos al software Mosquitto.

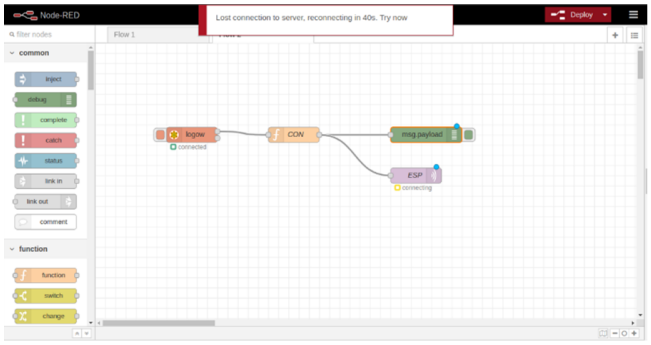

Figura 17: *Mensaje de Error de Conexión Node-Red*

V.2. Comprobación software Mosquitto

El sotware Mosquitto, al inicializarlo, en la interfaz mostrada en la Fig. 18, confirma el correcto funcionamiento del mismo, mediante la habilitación del puerto 1883, logrando recibir subscriptores al broker de manera local, evitando la necesidad de adquirir una cuenta "adafruit.io cloud", para el uso del software; también se aprecia en la Fig. 17, la versión de Mosquitto y mensajes de código. Ratificando nuevos dispositivos conectados al "broker", mediante el puerto 1883, confirmando lo anterior, mediante protocolo TCP/IP y la conexión de dispositivos, para recibir datos, como subscriptores al "broker"; reafirmando el funcionamiento del sistema de comunicación, entre el software Mosquitto y en este caso la placa de desarrollo ESP8266-12E, de manera exitosa.

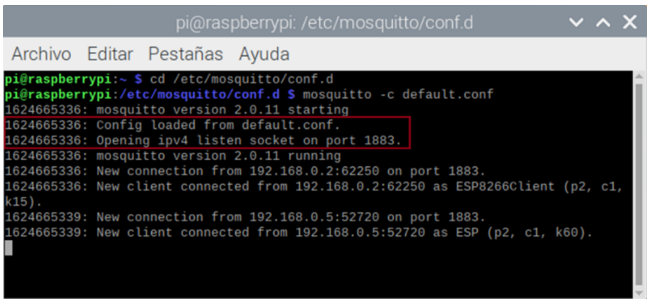

Figura 18: *Interfaz de Inicialización de Mosquitto y Subscriptores, mediante puerto 1883*

V.3. Comprobación ESP8266-12E

Concerniente a la placa de desarrollo ESP8266-12E, es viable realizar mediante el monitor serial del IDE de Arduino, la conexión al "broker", del software Mosquitto. producto de la impresión de mensajes declarados en el "sketch" del código programado y cargado a la placa de desarrollo mencionada, la Fig. 19, muestra mensajes de desconexión o error en la comunicación MQTT del dispositivo ESP8266-12E, hacia el "broker" del software Mosquitto. Realizando un análisis comparativo de la Fig. 16 y la Fig. 19, es posible apreciar y confirmar la conexión establecida entre la placa de desarrollo ESP8266-12E y el software Mosquitto, de manera correcta para el presente trabajo, además de valorar como una ventaja el programar mediante el IDE de Arduino, al existir la posibilidad de realizar análisis comparativos, mientras se elabora cualquier tipo de desarrollo, al usar el monitor serial del software mencionado.

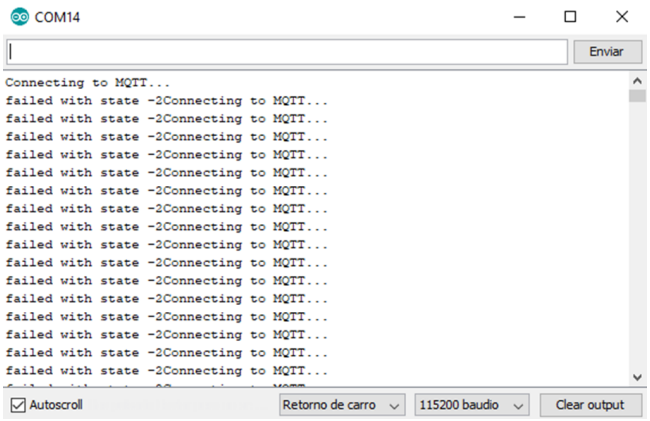

Figura 19: *Mensajes de Error en Monitor Serial del IDE de Arduino para ESP8266-12E*

VI. Conclusiones

La comunicación de 4 fases, concluye exitosamente para el envío de datos entre los diferentes dispositivos

y herramientas de software, resultando en un sistema de utilidad, de gama industrial, para generar cualquier tipo de control (atribuible al PLC Logo 8), concerniente a una adquisición asequible, es aplicable para la placa de desarrollo ESP8266-12E, la Raspberry Pi 4, las herramientas de software Node-Red y Mosquitto, debido a que son de fuente libre, representando un costo nulo para su respectiva adquisición con funcionamiento hasta ciertos parámetros.

Respecto a la Raspberry Pi4, resulta ser un dispositivo factible, debido a su bajo costo de adquisición en comparativa con un sistema Windows o MAC y además de una adecuada compatibilidad con las herramientas de software Node-Red y Mosquitto.

Relativo al software Mosquitto, al habilitar y utilizar el puerto TCP/IP 1883, genera un ahorro en el coste del sistema, lo anterior, debido al acceso libre para el uso del "broker" de Mosquitto, sin necesitar adquirir una cuenta "adafruit.io cloud", funcionando el software en red local, desempeñándose de manera aceptable, para cualquier sistema a operar con una red local. Además, para este tipo de sistema, el software Mosquitto, resulta ser la mayor solución para envío de datos, lo anterior, debido a la necesidad del sistema, concerniente al envío de una pequeña cantidad de bytes, resultado del control de cualquier variable a utilizar, mediante el PLC Logo 8, evitando enfoques informáticos clásicos como: servidores web, socket de red y entre otros más, liberando el ancho de banda y eficientando la comunicación.

Es posible encontrar desventajas en el sistema, refiriendo particularmente, el coste del PLC Logo 8, por ser un dispositivo de gamma industrial.

Agradecimientos

Dirigidos al TecNM, campus Instituto Tecnológico de Ciudad Guzmán, por proporcionar todo lo necesario para el desarrollo del presente trabajo, proporcionando desde sus instalaciones, equipo y herramientas para el desarrollo del presente trabajo.

A la empresa IBM, por desarrollar software de acceso libre para el desarrollo de proyectos con tecnología de vanguardia resultando ser las "open source" en una accesibilidad para el aprendizaje de las herramientas de software como lo son Node-Red y Mosquitto.

Referencias

[1] MOHAMMED Anwaruddin y MOHD ANAS Ali. «A Review On Raspberry Pi Based Industrial Process Monitoring And Control Using Modbus Protocol». En: *IJITR) INTERNATIONAL JOURNAL OF INNOVATIVE TECHNOLOGY AND RESEARCH Volume* 5 (2017), págs. 5483-5486.

[2] T. Hagino y N. O'Leary. *Practical Node-RED Programming: Learn powerful visual programming techniques and best practices for the web and IoT*. Packt Publishing, 2021. ISBN: 9781800207660. URL: https://books.google.com.mx/books?id=nlwlEAAAQBAJ.

[3] AI Marosan y col. «Creating an ethernet communication between a Simatic S7-1200 PLC and Arduino Mega for an omnidirectional mobile platform and industrial equipment». En: *IOP Conference Series: Materials Science and Engineering*. Vol. 968. 1. IOP Publishing. 2020, pág. 012022.

[4] Riffa Haviani Laluma y col. «Automation system of water treatment plant using raspberry Pi. 3 model B+ based on Internet of Things (IoT)». En: *2019 IEEE 13th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*. IEEE. 2019, págs. 72-76.

[5] Vatsal Gupta, Sonam Khera y Neelam Turk. «MQTT protocol employing IOT based home safety system with ABE encryption». En: *Multimedia Tools and Applications* 80.2 (2021), págs. 2931-2949.

[6] G.C. Hillar. *MQTT Essentials - A Lightweight IoT Protocol*. Packt Publishing, 2017. ISBN: 9781787285149. URL: https://books.google.com.br/books?id=4OEwDwAAQBAJ.

[7] Carlos Hervas Parra. «Análisis de rendimiento de protocolos de Publicación/Subscripción en comunicación con una Red de Sensores Inalámbricos Zigbee». Tesis doct. Universidad Nacional de La Plata, 2018.