INFORME DE ADSCRIPCIÓN

Nota: El presente documento **PUEDE ser modificado de acuerdo al estado de la beca**. Deberá ser adjuntado en formato PDF en la sección **AUTOARCHIVO** del SASPI con el denominador "**Documento de trabajo**". El nombre del archivo debe resumir y guardar correspondencia con el tipo de actividad, año y código si lo tuviera. Ejemplos:

Informe Técnico Beca EICyT 2020 - Paola Pomato
Informe Técnico Beca Doctoral CONICET 2020 - Pablo Pomato

Adscripto:	KIRSCHNER, Lucas Sebastián
Directores:	Mgter. Ing. Guillermo A. Fernández; Dr. Ing. Rubén Núñez
Co-director:	Dr. Ing. Fernando Botterón
Unidad Académica:	Facultad de Ingeniería
Área temática:	Ingenierías y Tecnologías
Código del Proyecto/Trabajo Acreditado:	16/I1083-PDTS
Adscripción – Duración:	1 año
Título del Plan de Trabajo:	Implementación experimental de comunicación de datos para el monitoreo de variables en microrredes eléctricas.

1. Exposición sintética de la labor desarrollada (no más de una página).

1.1 Análisis y selección del protocolo de comunicación entre las unidades UTM y UTR

Un protocolo de comunicación es el conjunto normas y reglas determinadas a cumplir por dos o más dispositivos que desean comunicarse entre sí. A continuación, se lleva a cabo la comparación entre tres protocolos de comunicación de uso industrial que tienen gran presencia en el entorno industrial y en base a las comparaciones realizadas es seleccionado uno para su implementación en el proyecto.

El protocolo Modbus [4] es ampliamente utilizado en la automatización industrial entre dispositivos inteligentes. Actualmente existen tres versiones del protocolo Modbus: ASCII, RTU y TCP. El objetivo de Modbus es la transmisión de información entre distintos equipos electrónicos conectados a un mismo bus, siguiendo el principio de comunicación maestro-esclavo (RTU o UTR en español) o bien el principio cliente-servidor (TCP). En ambos casos, en dicho bus existe un solo dispositivo maestro (cliente) y uno o varios equipos esclavos (servidores) conectados. Esto significa que un maestro Modbus (la unidad terminal maestra, UTM), que está conectada a uno o múltiples esclavos (unidades terminales remotas, UTR), solicitará información de los mismos y estos proporcionarán la información solicitada. La unidad maestra puede enviar un mensaje de difusión a todos sus esclavos o dirigirse a un solo dispositivo específico. Los esclavos responderán a todos los mensajes que se les enviaron individualmente, pero no responderán a los mensajes de difusión. Los esclavos Modbus tampoco pueden iniciar mensajes por sí mismos y solo pueden responder a las consultas de mensajes recibidos del maestro.

El protocolo de red distribuida versión 3.0 ó DNP3 (Distributed Network Protocol 3) [5], se trata de un protocolo público, abierto y optimizado, desarrollado para sistemas SCADA. Una estructura típica de DNP3 involucra una UTM que monitorea a todas las UTR del sistema y controla su comportamiento en

base a los datos recopilados de los mismos, en base al principio maestro-esclavo. Por lo general, el maestro es el que inicia la solicitud de datos y envía los comandos de control, a lo que la UTR correspondiente responde con la información apropiada o realizando la acción de control del equipo. La particularidad es que las UTR's también pueden enviar mensajes no solicitados para actualizar la estación maestra sobre algún cambio de estado importante del equipo en cuestión.

Otro protocolo de comunicación utilizado en la industria es PROFIBUS (Process Field Bus) [6], que constituye un estándar de red digital de campo abierto (bus de campo) encargado de la comunicación entre los sensores de campo y el sistema de control o los controladores. En la actualidad existen diferentes versiones: PROFIBUS-PA (*Process Automation*), PROFIBUS-FMS (*Fieldbus Message Specification*), PROFIBUS-DP (*Decentralised Peripherals*), todas ampliamente extendidas en el mundo y especialmente en el continente europeo. El protocolo derivado PA se orienta a las comunicaciones industriales. PROFIBUS también sigue el principio de comunicación maestro-esclavo, pero con la diferencia de que incluye un protocolo adicional que permite múltiples maestros en el mismo bus.

Para seleccionar uno de los protocolos de comunicaciones antes indicados, se recurre a las principales características de los mismos. Comparando los protocolos PROFIBUS y Modbus, tenemos que el primero es un protocolo muy robusto que fue diseñado para automatizar plantas enteras. Por su parte, Modbus es una solución fácil en aplicaciones donde existe una única UTM, mientras que PROFIBUS se destaca en implementaciones en donde están involucrados diferentes proveedores.

Debido al hecho de que PROFIBUS está orientado al uso como bus de campo (a niveles de dispositivos que controlan el proceso en el campo) y Modbus se orienta a la comunicación entre dispositivos en niveles superiores, se descarta el protocolo PROFIBUS frente a Modbus, dado que la comunicación entre la UTM y la UTR se lleva a cabo en niveles superiores al bus de campo.

Comparando ahora DNP3 y Modbus, se comentan a continuación las conclusiones que llevaron a seleccionar el protocolo Modbus para ser implementado en el sistema de comunicación entre la UTM y la UTR del sistema de supervisión que posee la microrred planteada en el proyecto de investigación:

- Tanto Modbus como DNP3 se basan en el principio de comunicación maestro-esclavo, pero en DNP3 las UTR también tienen posibilidad de enviar mensajes no solicitados para actualizar la estación maestra sobre algún cambio de estado importante del equipo. Esto da un mayor grado de complejidad al sistema de comunicación implementado mediante el protocolo DNP3, haciendo que la implementación de Modbus TCP sea más viable desde ese punto de vista.
- Ambos son protocolos con estándares abiertos, lo que significa que es posible la comunicación entre dispositivos de diferentes fabricantes.
- Aunque DNP3 es capaz de enviar más información a través de la red, para tipos de datos reducidos, Modbus es la mejor opción debido a su simplicidad de implementación.
- Modbus presenta velocidades de comunicación mayores que DNP3.

Atendiendo a las ventajas de Modbus sobre DNP3, para la comunicación entre la UTM y la UTR del sistema de supervisión, se escoge el protocolo Modbus.

Seguidamente se describen cuestiones relacionadas a la minicomputadora utilizada como UTR y su programación para comprobar la interacción con una aplicación SCADA que es ejecutada por la computadora de escritorio utilizada como UTM en el sistema de supervisión de la microrred.

1.2 Selección del lenguaje de programación de la minicomputadora

Para la implementación de la unidad terminal remota (UTR) del sistema de supervisión de la microrred, se pretende utilizar la minicomputadora Raspberry Pi 4 Modelo B (RPi4B) [7]. La Fig. 1 muestra a la RPi4B (con 4 GB de memoria RAM) utilizada en este trabajo, junto a la disposición de sus líneas de entrada/salida (pines GPIO).

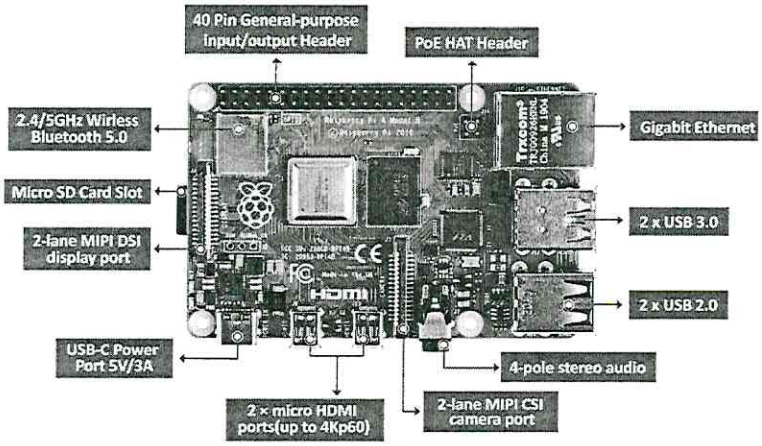


Fig. 1. Raspberry Pi 4 Model B.

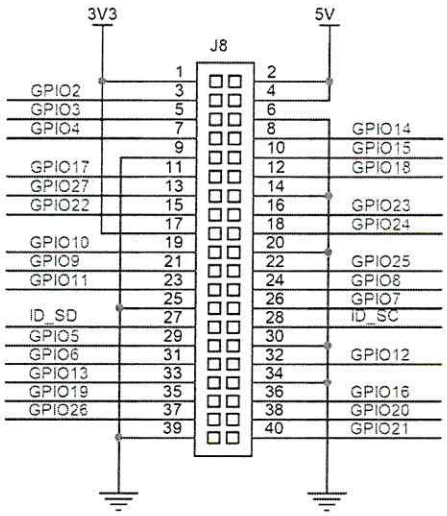


Fig. 2. Asignación de pines al conector GPIO.

La minicomputadora RPi4B tiene disponible 28 pines de Entrada/Salida de Propósito General (GPIO) a través de una tira doble de pines hembra estándar (2 x 20 pines), como se muestra en la Fig. 2. Además de poder usarse como entrada y salida controladas por software (con resistencias *pull-up* internas programables), los pines GPIO pueden operar en otros modos que son utilizados por bloques periféricos dedicados (como I2C, UART y SPI) que incorpora al RPi4B.

El sistema operativo seleccionado para operar la RPi4B en este proyecto, es Raspberry Pi OS. Se trata de un sistema operativo gratuito basado en Debian, optimizado para el hardware Raspberry Pi y además es el sistema operativo recomendado por el fabricante de RPi4B para uso normal en este dispositivo.

En el conjunto de paquetes de Raspberry Pi OS se incluye un entorno de desarrollo para Python 3 [8] llamado Thonny [9]. Python 3 se trata de un lenguaje de programación de alto nivel que presenta una sintaxis limpia con énfasis en la legibilidad, utilizando palabras clave estándares en inglés. Además, por ser de alto nivel, este lenguaje permite desarrollar programas complejos haciendo uso de librerías que facilitan la realización de los mismos. Por otra parte, el lenguaje Python es ampliamente difundido para

la programación de la RPi4B, encontrándose abundante información y numerosas herramientas desarrolladas por la comunidad. Este lenguaje incluye librerías que posibilitan el manejo de los pines GPIO y demás registros de la RPi4B, necesarios para controlar la comunicación con la UTM.

```
1  from gpiozero import LED, Button
2
3  led1 = LED(17)
4  button = Button(22)
5  led1.off()
6  while True:
7      if button.is_pressed:
8          led1.on()
9      else:
10         led1.off()
```

Fig. 3. Programa básico para manejo de pines GPIO.

Con la intensión de adentrarse en el manejo de los pines GPIO de la RPi4B y su relación con el entorno de desarrollo, se efectuaron pruebas sencillas. Las mismas consistieron en la lectura/escritura de pines establecidos como entrada/salida. La Fig. 3 muestra un ejemplo de un programa sencillo, donde mediante un pulsador asociado al pin GPIO22 establecido como entrada es posible encender/apagar un led conectado al pin GPIO17 de la RPi4B. Tanto el LED con su resistencia limitadora como el pulsador con su correspondiente resistencia de *pull-up* se conectaron haciendo uso de un protoboard. Para el desarrollo de este programa, es necesario utilizar la biblioteca GPIO Zero Python, la cual debe ser instalada de manera predeterminada en la imagen de escritorio que dispone Raspberry Pi OS.

1.3 Desarrollo de la aplicación SCADA para monitorización de variables a través de la PC

Para el desarrollo de la aplicación SCADA se utiliza el software especializado InduSoft Web Studio® [10], el cual posee una colección de herramientas de automatización que proporcionan todos los componentes básicos de la automatización para desarrollar interfaces hombre-máquina (HMI), aplicaciones SCADA y soluciones de instrumentación integradas. Es completamente configurable por el usuario y permite el monitoreo en tiempo real de variables a través de gráficos y otros objetos relacionados con las variables físicas que interesan medir. También permite realizar accionamientos, enviar y recibir información desde equipos remotos y automatizar diversas tareas de acuerdo a las necesidades del usuario.

Todos los datos intercambiados a través del protocolo Modbus (bits y registros), utilizado para comunicar a la UTM con la UTR del sistema de supervisión, deben ubicarse en la memoria de la aplicación del dispositivo servidor (dispositivo esclavo, UTR en este caso). Pero la dirección física en la memoria no debe confundirse con la referencia de datos. El único requisito es vincular la referencia de datos con la dirección física. Así, el protocolo Modbus TCP posibilita a las unidades clientes y servidor leer y escribir sobre una misma serie de tablas de datos, las cuales se indican en la Fig. 4, donde para cada una (entrada

discreta, bobina, registro de entrada o registros de retención), el protocolo permite la selección individual de 65536 elementos de datos.

Register Type	Description
0X	Coil Status
1X	Input Status
3X	Input Register
4X	Holding Register
ID	Slave ID Number

Fig. 4. Referencia de datos Modbus TCP.

Para la comunicación con la UTR a través de Modbus TCP, InduSoft Web Studio® proporciona una interfaz fácil de usar para configurar todas las direcciones de comunicación de la aplicación. En base a lo establecido en el protocolo, es necesario indicar el tipo de variable con el que se desea trabajar (entrada discreta, bobina, registro de entrada o registros de retención), el tipo de acción (lectura, escritura o ambos), la dirección IP de la UTR (RTU en inglés) específica y el puerto correspondiente. Para este caso, la dirección IP seleccionada es la 169.254.216.171, la cual es designada a la UTR. Por su parte, el puerto designado es el 5900 de la RPi4B, que está específicamente reservado para aplicaciones Modbus. Esto se indica en el campo Estación de las ventanas presentadas en la Fig. 5 y Fig. 6. Si el servidor fuese un dispositivo con Windows, el puerto designado sería el 502.

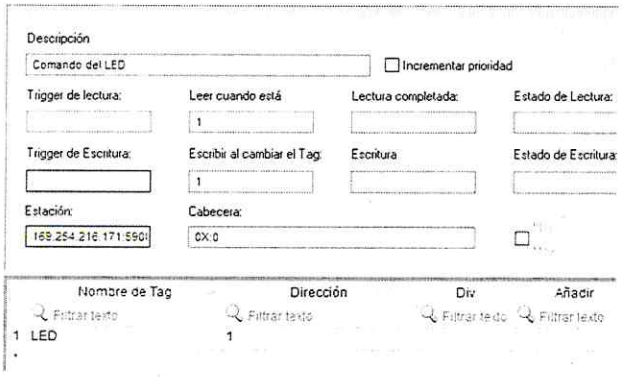


Fig. 5. Configuración de parámetros Modbus TCP para comandar el LED indicativo.

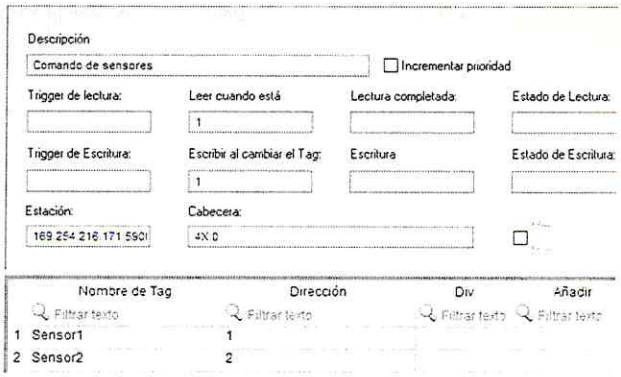


Fig. 6. Configuración de parámetros de Modbus TCP para comandar los sensores.

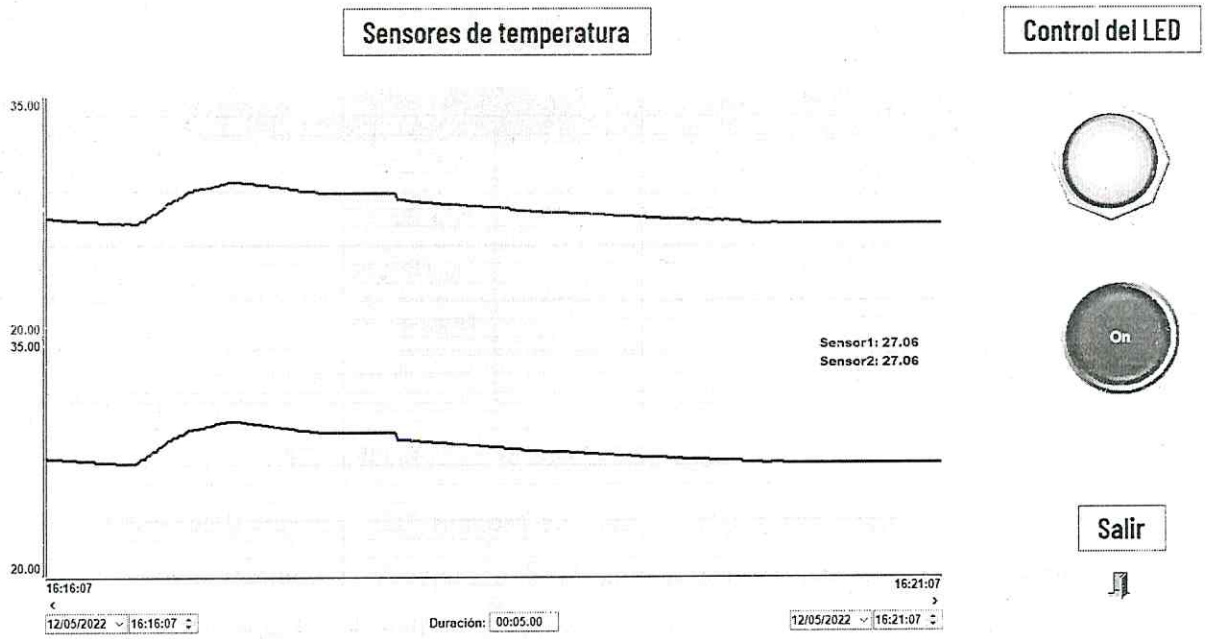


Fig. 7. Interfaz gráfica de la aplicación SCADA desarrollada.

Para las pruebas llevadas a cabo se designó el primer espacio de la tabla de datos de bobinas (valor 0X:0 en el campo Cabecera de la Fig. 5 y Fig. 6) para almacenar el estado (cero o uno, apagado o encendido) de un LED indicativo asociado a un pin GPIO de la RPi4B, el cual puede ser controlado con el botón de color verde en la aplicación SCADA cuya interfaz se muestra en la Fig. 7. De igual manera se utilizaron los dos primeros espacios de la tabla de datos de registros de retención (valor 4X:0 en el campo Cabecera de la Fig. 5 y Fig. 6) para guardar el valor digital correspondiente a dos entradas de la RPi4B, cuya variación temporal se muestra en las gráficas de la interfaz.

1.4 Desarrollo del programa de la minicomputadora (UTR) para interactuar con la aplicación SCADA de la UTM

Para el manejo del protocolo Modbus TCP desde la UTR, se optó por la utilización de la librería pyModbusTCP [11], la cual contiene módulos para el manejo de clientes y servidores Modbus TCP, un módulo especial llamado "utils" que proporciona algunas funciones útiles para la manipulación de datos Modbus TCP y un módulo llamado DataBank, el cual permite administrar las tablas de datos del protocolo Modbus TCP.

El programa desarrollado para la UTR consiste en la creación de un servidor Modbus TCP mediante el módulo ModbusServer de la librería pyModbusTCP. Se designa la dirección IP estática 169.254.216.171 y se establece la comunicación mediante el puerto 5900 de la RPi4B.

Habilitado el protocolo 1-Wire [12] de la RPi4B mediante la librería "w1thermsensor", se crearon dos objetos para el manejo de dos sensores de temperatura DS18B20 [13]. Haciendo uso de un protoboard, los sensores fueron conectados a la alimentación de 3,3 V, GND y el GPIO4 de la RPi4B, utilizando una

resistencia de *pull-up*, al bus de comunicación compartido por ambos sensores. La temperatura registrada por los sensores es almacenada en los dos primeros registros de retención del protocolo Modbus TCP, haciendo uso del módulo DataBank correspondiente a la librería pyModbusTCP.

En la bobina 1 se guarda el estado del pin GPIO17 de la RPi4B establecido como salida y configurado mediante el método LED de la librería "gpiozero". El mismo es conectado a un LED ubicado en el protoboard, con la correspondiente resistencia limitadora. Para el control del LED, se incluye en el mismo protoboard un pulsador con su correspondiente resistencia de *pull-up*, cuyo estado se asocia al pin GPIO22 de la RPi4B establecido como entrada.

Seguido a la definición inicial de variables y la creación de objetos, el programa permanece en un bucle de manera indeterminada. Durante este bucle, el programa efectúa la actualización de los valores de temperatura almacenados en los registros de retención.

La medición de temperatura con los sensores DS18B20 produce en la ejecución del programa un retardo de aproximadamente un segundo. Para solucionar este inconveniente, se utiliza la librería "threading", la cual permite la creación de hilos de ejecución en paralelo. De esta manera, todo el proceso de medición de temperatura y el almacenamiento en los registros de retención de Modbus TCP se realiza en el hilo secundario.

Debe tenerse en cuenta que cada registro de retención tiene un ancho de palabra para cada dato de 16 bits y en el mismo debe cargarse un valor entero en complemento a 2. Por lo tanto, el rango de representación se extiende desde -32768 hasta +32767. Para llegar a una relación de compromiso entre precisión y rango de valores representados, es considerado solamente dos dígitos decimales. Lo anterior es obtenido multiplicando por 100 el valor a ser transmitido desde la UTR (Fig. 8a), teniendo en cuenta que en la UTM debe efectuarse el proceso inverso (Fig. 8b).

```
DataBank.set_words(0,[(sensor1.get_temperature())*100])
DataBank.set_words(1,[(sensor2.get_temperature())*100])
```

(a)

	Nombre de Tag	Expresión
	Filtrar texto	Filtrar texto
1	led_toglea	Toggle(LED)
3	Sensor2_div	Sensor2/100
2	Sensor1_div	Sensor1/100

(b)

Fig. 8. Acondicionamiento de valores decimales transmitidos: (a) Programa de la UTR; (b) Programa de la UTM.

Mediante la librería "tkinter" es creada una interfaz gráfica para la UTR. Esto es para considerar el caso en que en esta unidad se encuentre disponible un monitor. La interfaz mencionada es presentada en la Fig. 9, donde la misma muestra los valores leídos de los sensores, el estado de la conexión con la UTM, el estado del LED indicativo y un botón para cambiar su estado.

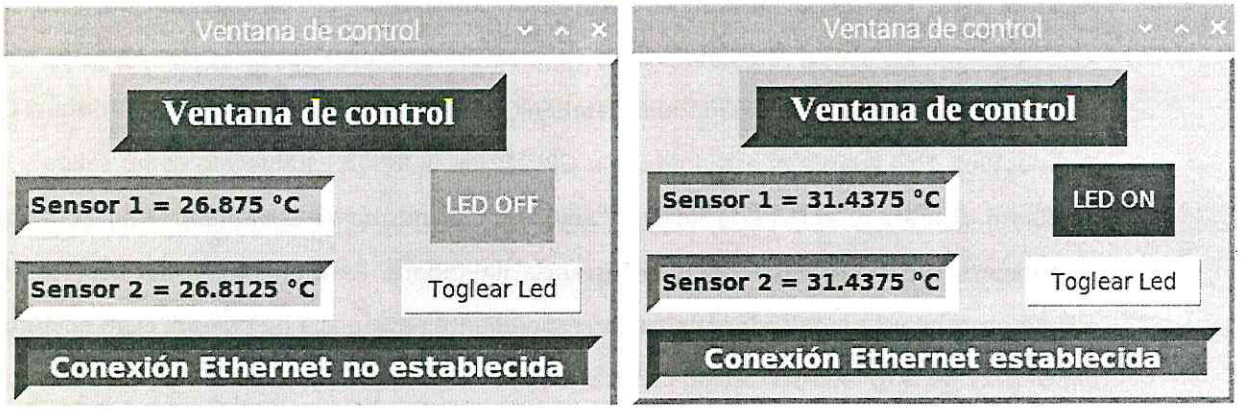


Fig. 9. Interfaz gráfica de la UTR.

Para el desarrollo del programa de la UTR se tiene la consideración de que la misma puede operar con o sin la conexión a la UTM. Para ello, continuamente la UTR intenta conectarse a la UTM. En caso de que lo consiga, se establece la comunicación, siendo posible para la UTM (la cual ejecuta la aplicación SCADA) extraer los datos deseados. La librería pyModbusTCP no cuenta con una función para detectar la desconexión de la UTM, por ello se incluye la librería "ping3", de la cual se utilizan funciones para testear el estado de la conexión, determinando que la UTM ya no se encuentra conectada en caso de que no sea posible hacer ping a la IP especificada para dicha conexión.

2. Objetivos alcanzados.

En base a los resultados presentados, puede observarse que se cumplió con el objetivo de establecer la comunicación de datos entre las unidades terminal maestra (UTM) y remota (UTM) para el sistema de supervisión de una microrred eléctrica. Para la comunicación de datos mencionada se seleccionó el protocolo Modbus TCP/IP, la cual fue escogida efectuando una comparación entre diferentes protocolos de comunicación industrial.

Mediante un software dedicado se desarrolló una aplicación SCADA básica, la cual posibilitó la supervisión y control de variables de manera remota a través de una computadora personal, que ofició como unidad terminal maestra.

La unidad terminal remota fue implementada con una minicomputadora Raspberry Pi 4 Modelo B, mediante la cual se controlan las variables de campo a partir de las ordenes que llegan desde la unidad terminal maestra.

Los resultados expresados nos llevan a concluir que es posible implementar el sistema de supervisión de la microrred propuesta en el proyecto de investigación, donde la unidad terminal maestra está constituida por una computadora personal de escritorio (o notebook) que ejecuta la aplicación SCADA y se comunica a través de un protocolo industrial de comunicación de datos con una unidad terminal remota constituida por una minicomputadora RPi4B, encargada de ejecutar un algoritmo de supervisión y gestión que opere en base los valores de las variables medidas en la microrred.

3. Métodos y técnicas empleados.

Se comienza con el estudio de la computadora RPi4B, analizando su funcionamiento y el modo de operación de sus pines GPIO a través del lenguaje de programación Python.

Seguidamente se estudian las características del protocolo Modbus TCP y su implementación mediante el uso de una librería de Python a fin. Se efectúan las configuraciones necesarias para la recepción y envío de datos desde la UTR hacia la UTM.

A continuación, se analiza el software con el cual se pretende desarrollar la interfaz de usuario de la aplicación SCADA. Se lleva a cabo la creación de un programa sencillo a través del cual el usuario tiene control desde la UTM sobre variables asociadas a la UTR.

Para finalizar, se lleva a cabo la implementación de la comunicación bidireccional, a través de Modbus TCP, entre las unidades maestra y remota.

4. Bibliografía consultada.

[1] Fernando Botterón, «Bombeo de agua con energías renovables, almacenamiento de energía y conexión a la red para pequeñas huertas rurales comunitarias: estudio, diseño y puesta en funcionamiento».

[2] Alejandro G. Maxit, Guillermo A. Fernández, Fernando Botterón, «Avances en el desarrollo de un sistema de supervisión de una microrred utilizada para el bombeo de agua en huertas comunitarias rurales».

[3] A. Rodríguez Penin, Sistemas SCADA (2a. ed.)., Barcelona, Spain: Marcombo, 2008.

[4] Modbus, [En línea]. Available: https://modbus.org/. [Último acceso: 02 12 2022].

[5] DNP, [En línea]. Available: https://www.dnp.org/. [Último acceso: 02 12 2022].

[6] PROFIBUS & PROFINET International (PI), [En línea]. Available: https://www.profibus.com/. [Último acceso: 02 12 2022].

[7] Raspberry Pi Foundation, «Raspberry Pi 4 Model B datasheet,» [En línea]. Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf. [Último acceso: 02 12 2022].

[8] Python TM, «Python,» [En línea]. Available: https://www.python.org/. [Último acceso: 02 12 2022].

[9] Thonny, «Thonny,» [En línea]. Available: https://thonny.org/. [Último acceso: 02 12 2022].

[10] OMRON, «Indusoft Web Studio®,» [En línea]. Available: https://www.cpi.com.ar/productos/indusoft-web-studio/. [Último acceso: 02 12 2022].

[11] «pyModbusTCP's documentation,» [En línea]. Available: https://pymodbustcp.readthedocs.io/en/latest/index.html. [Último acceso: 02 12 2022].

[12] Analog Devices, «Guide to 1-wire communication,» [En línea]. Available: https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1796.html. [Último acceso: 02 12 2022].

[13] Maxim Integrated, «DS18B20 datasheet,» [En línea]. Available: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf. [Último acceso: 02 12 2022].

[14] Raspberry Pi Foundation, [En línea]. Available: https://www.raspberrypi.com/. [Último acceso: 02 12 2022].

5. **Resultados obtenidos expresada en indicadores de CyT: trabajos publicados, en prensa, presentaciones a reuniones científicas, etc. (colocar referencia a material digital).**

Kirschner, Lucas Sebastián; Mayer, Cintia Liset; Botterón, Fernando; Fernández, Guillermo Alfredo. "Sistema para supervisión remota de variables de proceso". Jornadas de Investigación y Desarrollo Tecnológico, Extensión, Vinculación y Muestra de la Producción, XII JIDeTEV. Oberá, Misiones, agosto de 2022.

6. **Obstáculos y dificultades halladas durante el desarrollo del plan de trabajo.**

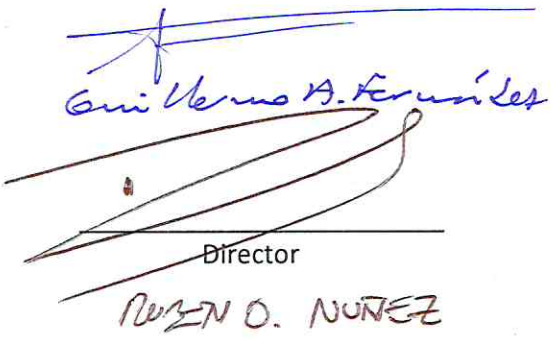
El principal obstáculo hallado durante el desarrollo del plan de trabajo fue la falta de conocimiento sobre el protocolo de comunicación Modbus TCP. Agregado al hecho de que no se contaba con experiencia en el manejo de la computadora Raspberry Pi utilizada, fue un gran desafío lograr la transmisión de datos desde la UTR hacia la UTM.

7. **Avance académico durante el período de la adscripción (exprese en porcentaje):** 25,64 %

Becario
Kirschner Lucas.

Evaluación del Director y Co-director (validación por sistema)

Evaluación del becario: SATISFACTORIO



Director
RUBEN O. NUÑEZ

Co-Director
Botterón Fernando