Didactic Prototype for Teaching the MQTT Protocol Based on Free Hardware Boards and Node-RED

A. Andrada, R. Murdocca, C. Sosa, and J. Dondo

***Abstract*—Due to the constant evolution of the technologies related to the Internet of Things and the great expansion of the protocols and devices that possibilities their use, there is a need to enforce and to approach the study of its concepts and applications to the students of electronic engineering and technical university careers. In this work, a weather station is designed and built based on low-cost technologies, whose data can be visible through an online platform designed for this purpose. The weather station is oriented to be used in laboratory practices and for teaching of subjects related to IoT, sensing platforms, communication platforms, etc. The general objective of the low-cost weather station platform is to provide a didactic, portable and simple design for replication in classroom in order to provide a whole system from the beginning to the students. Thanks to the use of this low-cost platform it was possible to perform better approach to this technology obtaining great improvements in the learning process of the students.**

***Index Terms*—*Arduino UNO*, IoT, M2M, MQTT, Node-Red.**

I. Introducción

En la experiencia de impartir prácticas de laboratorio a los alumnos de carreras de ingeniería, profesorado y tecnicatura en electrónica, se detectaron ciertas dificultades al momento de explicar el funcionamiento de protocolos relacionados con el Internet de las Cosas (*Internet of Things* – IoT*)* [1] y de cómo los sistemas aislados pueden aprovechar sus propiedades para que otros usuarios accedan a los datos obtenidos, sin necesidad de estar en la misma red local.

La importancia de que estos temas puedan ser comprendidos por los alumnos radica en que las tecnologías emergentes deben integrar su formación académica.

En este trabajo se propone el diseño de un prototipo, adaptado a los requerimientos del ámbito educativo universitario y de bajo costo, para soporte de la enseñanza del protocolo *Message Queuing Telemetry Transport* (MQTT) [2] en las prácticas de laboratorio. El sistema consta de una estación meteorológica que publica sus datos en una plataforma online, ver Fig. 1.

Astri Edith Andrada Tivani, Roberto Martín Murdocca, Carlos Federico Sosa Paez y Julio Dondo Gazzano son docentes investigadores del Departamento de Electrónica perteneciente a la Facultad de Ciencias Físico Matemáticas y Naturales de la Universidad Nacional de San Luis, Ejército de los Andes 950, CP: 5700, San Luis Argentina (e-mail: aeandrada@unsl.edu.ar).

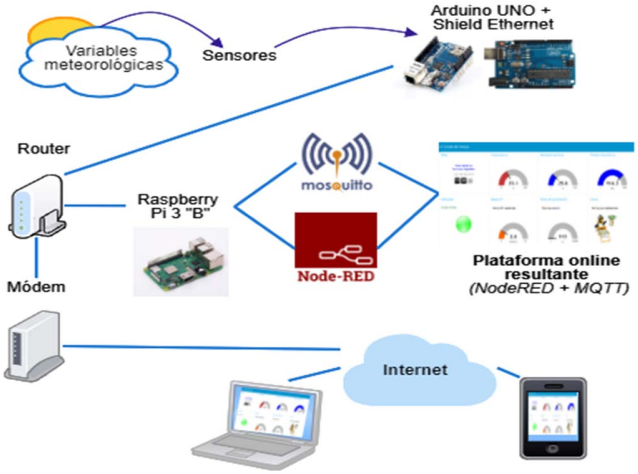

Fig. 1. Diagrama general del prototipo.

Dentro de la oferta de protocolos que se utilizan en el Internet de las Cosas, además del protocolo MQTT, existe *Constrained Application Protocol* (CoAP) [3].

Se realizaron pruebas con ambos protocolos sobre la placa Raspberry Pi 3. De dichas evaluaciones se obtuvieron resultados satisfactorios en la comunicación de la red local, en cuanto a las realizadas desde la red global con MQTT se experimentaron menos dificultades que con CoAP.

Existen diversas plataformas de IoT, se probó en primer lugar Node-RED [4] y luego Sofia2 [5]. Con Sofia2 se obtuvo una estética muy buena, gran fluidez de datos, pero para manejo de ciertos dispositivos y tener acceso a todo su potencial se requiere adquirir una licencia paga, y esto vulnera el principio de que el prototipo se pueda utilizar en un ambiente educativo donde su replicación se ajusta a un determinado presupuesto.

Teniendo en cuenta lo expresado en párrafos anteriores, sumadas a las pruebas de performance de otros autores [6] y una investigación realizada [7] se utiliza el protocolo MQTT y la herramienta de programación de flujos Node-RED.

En el caso de los sistemas embebidos empleados se eligieron placas de hardware libre y sensores de bajo costo.

Este paper está organizado de la siguiente manera: en la siguiente sección se muestran algunos trabajos relacionados, luego se describe la estación meteorológica con su respectivo hardware y firmware, seguidamente el protocolo de comunicación MQTT, después la herramienta Node-RED, las redes, resultados y finalmente las conclusiones.

II. Trabajos Relacionados

Similar a lo expuesto en el trabajo de Chanthakit y Rattanapoka [8] en este trabajo el *firmware* del dispositivo que maneja los sensores, lee los datos y los envía (publica) al *broker* MQTT. Además, la herramienta Node-RED se subscribe a los diferentes *topics* para recibir los datos desde el *broker* MQTT. Sin embargo, la diferencia básica con el presente *paper* está en la implementación del *broker* MQTT, ya que se utilizó un *broker* online, y para el dispositivo didáctico se instaló en una placa de hardware libre Raspberry Pi. El principal motivo que llevó a la utilización de dicho sistema embebido es que el alumno sea capaz de conocer e instalar un broker y pueda observar los diferentes estados de la comunicación.

Kodali y Anjum [9] plantearon la utilización del módulo WiFi ESP8266 para el manejo de sensores, lo que significa que la comunicación del publicador se efectúa en forma inalámbrica. En el presente trabajo se utilizó un shield Ethernet para el publicador Arduino UNO. La justificación se basa en que el lugar de emplazamiento de las estaciones meteorológicas suele ser en techos de edificios o a la intemperie. Se utilizó PoE (Power over Ethernet), para evitar la conexión a la red de 220V y se permitió aprovechar los beneficios de la red cableada con respecto a la inalámbrica. Esto hace que el alumno tenga un acercamiento a las problemáticas que suelen presentarse en casos de implementación fuera del ámbito académico.

Tal como lo trabajaron Rajalakshmi y Shahnasser en [10] la herramienta Node-RED se instaló en la placa Raspberry Pi. Se utilizaron nodos existentes en la versión utilizada y otros que se encuentran almacenados en GitHub.

III. Estación Meteorológica

La estación meteorológica es una forma de aplicación de la ciencia y tecnología para conocer y predecir las condiciones climáticas en una ubicación particular. Se diseña para recoger datos cuantitativos sobre las condiciones climáticas [11]. Para poder tomar muestras fehacientes del fenómeno climático debe estar emplazada a la intemperie. Como el prototipo de este trabajo tiene fines educativos, se tuvo en cuenta instalar la estación en una maqueta funcional que permitiera su traslado sin inconvenientes.

El equipo diseñado permite medir las siguientes magnitudes físicas: temperatura, humedad relativa, presión atmosférica, nivel de iluminación, presencia de precipitaciones e intensidad de luz ultravioleta. Se emplearon diversos sensores y la plataforma embebida Arduino UNO con un *shield* Ethernet (o placa de expansión Ethernet).

Primero se realizaron las pruebas pertinentes con cada sensor individual y la plataforma embebida. Luego se los agrupó, con sus correspondientes interfaces e implementación de protocolos, para lograr estabilidad y probar el correcto funcionamiento del sistema. Cabe destacar que, al utilizar estos dispositivos, el prototipo incrementa su funcionalidad e integra mayor cantidad de prácticas de laboratorio.

A. Hardware

A continuación, se dará una breve descripción de los elementos de hardware utilizados en la estación meteorológica.

1) Placa Arduino UNO

Arduino Uno es una plataforma embebida basada en el microcontrolador ATmega328P. Cuenta con 14 pines digitales de entrada / salida (de los cuales 6 se pueden utilizar como salidas PWM), posee 6 entradas analógicas, un cristal de cuarzo de 16 MHz, conexión USB, conector de alimentación y un botón de reinicio. La placa posee todo el hardware de soporte necesario para el microcontrolador; basta con conectarlo a una PC con un cable USB para efectuar su programación y proveerle alimentación a través de una fuente de corriente continua o una batería.

Se eligió la plataforma Arduino Uno porque no se requiere un procesador de alto desempeño para la etapa de adquisición de datos de los sensores de la estación meteorológica.

Las tareas de la placa fueron el procesamiento y envío de datos al broker MQTT.

2) Shield Ethernet W5100

Es el dispositivo utilizado para que Arduino UNO tenga la capacidad de conexión a la red de área local.

3) Sensores

- Sensor detector de lluvia YL-83 (módulo Raindrop)
- Sensor de presión atmosférica BMP280
- Sensor de luz ultravioleta ML8511
- Sensor de nivel de iluminación BH1750
- Sensor de temperatura y humedad relativa DHT22

4) Power over Ethernet, PoE

Es el dispositivo que se utiliza para prescindir de la conexión a 220V en la zona que se encuentra emplazada la estación meteorológica.

La alimentación a través de Ethernet (PoE) es una tecnología que incorpora alimentación eléctrica a una infraestructura LAN estándar. Permite que la alimentación eléctrica se suministre a un dispositivo de red usando el mismo cable que se utiliza para la conexión de red. Elimina la necesidad de utilizar tomas de corriente en las ubicaciones del dispositivo alimentado y permite una aplicación más sencilla de los sistemas de alimentación ininterrumpida (SAI) para garantizar un funcionamiento las 24 horas del día, 7 días a la semana.

Se utiliza para el proyecto un Inyector POE pasivo para llevar energía hacia la zona de los sensores y Arduino UNO.

B. Firmware

El *software* que maneja físicamente al *hardware* de la placa Arduino UNO se programó basado en el modelo de Máquina de Estados Finitos (FSM por acrónimo en inglés) [12]. Con esto se pretende facilitar la lectura del código, mejorar la detección de errores y su respectiva corrección. El diagrama de estados de la implementación realizada se puede observar en la Fig. 2.

La placa realiza las siguientes tareas dentro del prototipo:

- *Lectura de valores provenientes de los sensores*
- *Procesamiento de los datos medidos*
- *Envío de datos procesados a través del protocolo MQTT*

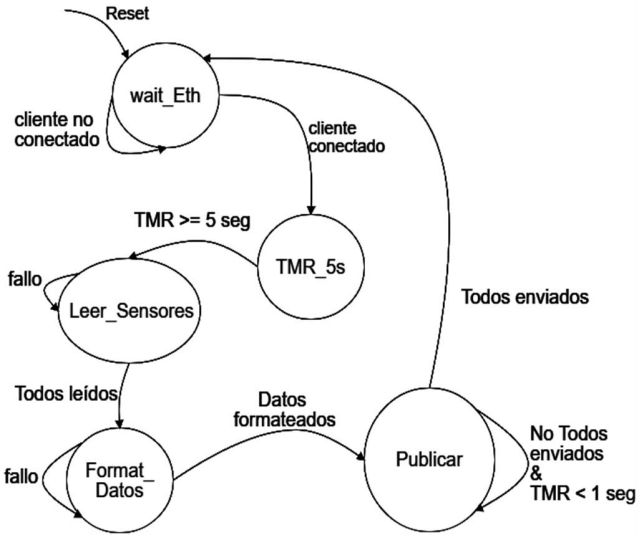

Fig. 2. Diagrama de FSM implementada en Arduino UNO.

Los estados involucrados en la implementación de la máquina de estados finitos fueron:

- *wait_Eth:* se realiza la conexión con el *broker* MQTT.
- *TMR_5s:* una vez establecida la conexión se realiza una demora de 5 segundos.
- *Leer_Sensores:* al cumplirse los 5 segundos, se realiza la lectura de los valores captados por los sensores.
- *Format_Datos:* obtenidos los datos se realiza su procesamiento. Se debe adaptar el formato de las mediciones recibidas, para que puedan enviarse a la placa Raspberry Pi 3 por el protocolo MQTT. Es necesario convertir el valor numérico en otro con características adecuadas (números de decimales, promedios, mapeo de datos de un grupo a otro). Finalmente, se debe convertir a una cadena de caracteres, y para ello se emplea la función char* dtostrf (double _val, signed char_width, unsigned char_prec, char*_s).
- *Publicar:* una vez realizado el procesamiento, se efectúa la publicación de los valores.

Finalizado el envío de todos los datos, se regresa al estado *wait_Eth*.

Para la utilización de sensores, conexión a red local y publicación de valores con protocolo MQTT se utilizaron funciones de las librerías de la comunidad Arduino y fabricantes de los dispositivos.

IV. Protocolo de comunicación MQTT

MQTT son las siglas de *Message Queue Telemetry Transport*. Es un protocolo ideado por IBM, y liberado para que se pueda utilizar enfocado a la conectividad Machine-to-Machine (M2M). [13]

Está enfocado al envío de datos en plataformas de hardware limitado y en redes con ancho de banda de alta latencia. Su flexibilidad hace que pueda soportar varios escenarios para dispositivos y servicios de IoT.

El protocolo posee tres niveles de calidad de servicio QoS. Para la finalidad del prototipo, basta con el nivel 0, donde el mensaje es entregado como máximo una sola vez.

La arquitectura de MQTT se puede dividir en dos componentes principales: *broker* y clientes.

- *El broker MQTT:* es el elemento encargado de gestionar la red y transmitir los mensajes, el servidor.
- *Los clientes MQTT:* el cliente puede ser un publicador, subscriptor o un publicador/suscriptor (siempre y cuando el hardware y/o software se lo permita). Siempre establecen conexión con el *broker*.

La comunicación se articula a través de *topics*. Éstos son identificadores que definen el contenido de los mensajes. Un emisor y un receptor están comunicados siempre y cuando estén suscritos a un *topic* común. El *broker* clasifica los mensajes a través de los *topics* y con ellos distingue los mensajes que corresponden a cada cliente.

En resumen, todo aquel que quiera publicar un mensaje MQTT es responsable de clasificarlo en un *topic* concreto y los clientes suscritos a ese *topic* recibirán esos mensajes.

Los *topics* en el protocolo MQTT se organizan según una estructura jerárquica en árbol (*topic tree*), utilizando el carácter "/" para formar un *topic* de varios niveles.

La arquitectura toma la forma que se observa en la Fig. 4, donde se puede ver de forma clara que una variación de una magnitud física es captada por un sensor. El dato obtenido se envía a C1. Éste procesa el dato y lo publica en un *topic.* El mensaje es recibido por B. Luego B reenvía el mensaje a los clientes C2, C3 y C4, siempre y cuando estén suscripos al *topic.*

El *bróker*, identificado con la letra B, se implementa en la computadora de placa simple Raspberry Pi 3. En el caso de los clientes que realizan publicaciones, el diseño preliminar preveía que fueran los sensores. Sin embargo, para disminuir costos, se eligieron dispositivos que no poseen inteligencia suficiente para realizar una publicación de datos o para conectarse a la red local por sí mismos. Por tal motivo, el cliente publicador es Arduino UNO, se puede identificar como C1. En el caso de clientes suscriptores se tienen 3 clientes: dos dentro de la red local, C2 y C3, y uno en la red global, C4.

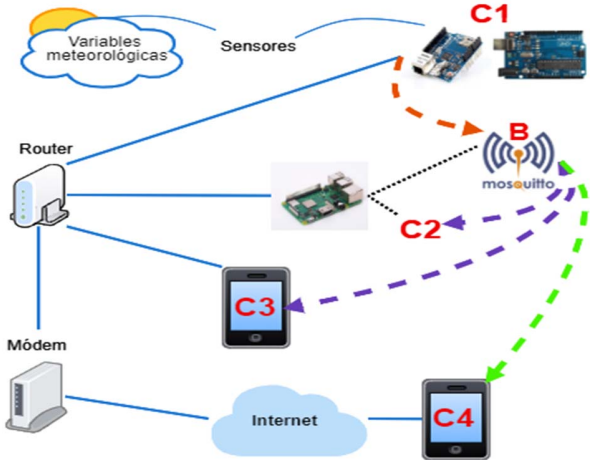

Fig. 3. Arquitectura protocolo MQTT en el trabajo.

Como en la placa Raspberry Pi 3 se puede tener alojado el servidor y clientes del protocolo, se utilizó tanto para B como para C2.

Para los clientes C3 y C4 se emplearon teléfonos celulares, utilizando una aplicación para Android llamada *MyMQTT*.

Es importante destacar que las primeras pruebas se realizaron en su totalidad en la red local. Finalizada esta etapa, se accedió a los datos por la red global.

A. Broker MQTT

En este proyecto se utiliza el *broker Mosquitto,* instalado en la plataforma embebida Raspberry Pi 3, modelo B.

Eclipse *Mosquitto* es un intermediario de mensajes de código abierto que implementa el protocolo MQTT en versiones 3.1 y 3.1.1. Es parte de la Fundación Eclipse y es un proyecto de iot.eclipse.org. Siendo altamente portátil, está disponible para una amplia gama de plataformas. [14]

Para instalarlo en la placa se siguió un instructivo provisto por el sitio web oficial, a través de acceso remoto por medio de SSH. Una vez instalado *Mosquitto* en la Raspberry Pi, se inició la actividad del *broker* con el comando *mosquitto*. Otra opción es *sudo /etc/init.d/mosquitto start*.

- *Raspberry Pi 3B*

La Raspberry Pi es un ordenador de bajo costo del tamaño de una tarjeta de crédito, desarrollada en Reino Unido por la fundación Raspberry Pi, se utiliza en el ambiente educativo para introducir a la electrónica. El uso de esta placa le brinda un buen nivel de flexibilidad y simplicidad al proyecto en comparación con otros entornos de desarrollo.

Para tener en cuenta en este trabajo se utiliza el sistema operativo *Raspbian*, que es el sistema operativo oficial para todos los modelos de Raspberry Pi. La versión utilizada es la de marzo 2018 y se trata de una distribución basada en *Debian Strech* (*Debian* 9.4).

V. NODE-RED

Node-RED es una herramienta de programación basada en flujo, de tipo *open-source*, desarrollada originalmente por el equipo de IBM's Emerging Technology Services y, en la actualidad, forma parte de la JS Foundation.

La programación basada en flujo es una forma de describir el comportamiento de una aplicación como una red de cajas negras o "nodos". Se proporciona una extensa cantidad y tipos de nodos cuando se realiza la instalación, organizados en paletas. En caso de necesitar funcionalidades extra existe la posibilidad de programar nodos nuevos, modificar los existentes e instalar nodos creados por otros usuarios.

Para instalar y utilizar la herramienta Node-RED en la plataforma Raspberry Pi 3 se puede utilizar el instructivo de la página web oficial. Se utilizó la versión 0.18.4.

La base para utilizar Node-RED es tener instalado y actualizado Node.js[1] y npm[2]. En este trabajo se utiliza la versión v8.11.1 de Node.js y 5.6.0 de npm.

Por recomendación, debido a posibles limitaciones de memoria disponible en la placa, se ejecuta Node-RED con el comando *node-red-pi*. Esto permite que se proporcione un argumento adicional que establezca en qué punto Node.js comenzará a liberar la memoria no utilizada. La opción es *max-old-space-size*, por lo que la ejecución del comando completo es *node-red-pi --max-old-spacesize=256*. El valor 256 hace referencia a 256Mb y puede modificarse.

Finalizada la instalación y la ejecución de la herramienta, para programar se utiliza un navegador web. Puede utilizarse tanto en la placa Raspberry Pi como en otro dispositivo que tenga acceso a la red local.

A continuación, se mencionan los flujos que se trabajaron en general, ya que dependiendo la variable se introdujeron ligeras modificaciones:

- Flujo para visualización de datos (nodo mqtt suscriptor, nodo de gráfica)
- Flujo para gráficas de evolución de datos (nodo mqtt subscriptor, nodo de gráfica evolutiva, nodo de conversión de tipo de dato, nodo de función valor máximo, nodo de función valor mínimo, nodo de función valor promedio, nodo muestra de texto)
- Flujo para almacenamiento de datos (nodo mqtt suscriptor, nodo de función para dar formato a medida, nodo de marca temporal, nodo de formato de fecha/hora, nodo para dar formato deseado a la fecha y la hora, nodo de para almacenar dato en archivo, nodo botón para iniciar borrado de archivo, nodo de eliminación de archivo)
- Flujo para audio (nodo mqtt suscriptor, nodo de función para dar formato a datos, nodo link, nodo de audio de salida)
- Nodos con funciones específicas (nodo para utilizar código HTML)

El resultado de la utilización de los diversos nodos y flujos de datos es una interfaz sencilla, accesible y funcional para el usuario final. Posee la adaptación automática para cualquier tamaño de pantalla, sin que el programador deba preocuparse por ello.

VI. REDES

Por la naturaleza intrínseca del prototipo, es necesario tener en cuenta ciertos detalles de las redes involucradas, tanto local como global.

En la red local se utiliza el router TP-Link, TL-WR740N. Se realiza enrutamiento estático y dinámico. Sólo se utiliza estático en el caso de la placa Raspberry Pi 3, ya que es necesario reconocer inequívocamente al servidor de la herramienta Node-RED y al *broker* MQTT. Pero, no es suficiente para determinar cuál de ellos dos ha de recibir el dato proveniente de otro punto de la red, por lo que se necesita utilizar el puerto de cada aplicación: 1883 para MQTT y 1880 para Node-RED (ambos son los que vienen por *default* para comunicaciones sin encriptación de datos).

Para acceder a programar con la herramienta Node-RED, se coloca simplemente el número IP asignado a la placa Raspberry Pi 3, seguido del número de puerto correspondiente: IP:1880. Para poder observar la plataforma resultante se debe colocar "/ui" (ui es el acrónimo de user interface) luego del número de puerto, esto es: IP:1880/ui. De esta manera, se puede utilizar la

[1] Node.js es un servidor capaz de ejecutar código JavaScript

[2] (Node Package Manager) Gestor de módulos y aplicaciones para Node.js

herramienta desde un navegador web dentro de la placa u otro dispositivo que esté conectado en la red local.

Para poder ingresar desde la red global es necesario conocer el IP público con el que figura el servidor, como éste puede variar en el transcurso del tiempo, porque lo suele cambiar el proveedor, se configura, en el router, un servicio DNS Dinámico, que asocia una dirección IP variable a un nombre de dominio fijo.

Para el proyecto se escogió como proveedor del servicio DDNS a No-IP. Se necesita tener una cuenta registrada en el sitio NoIP, y, además, se debe haber establecido la dirección que se utilizará para poder ingresarla en la configuración.

Para lograr fehacientemente la comunicación, se realiza el direccionamiento del tráfico proveniente de la red global a los servidores ubicados en la red local, por medio del router.

V. Resultados

A. El prototipo en la Actividad Docente

El prototipo se empleó en el dictado de clases de laboratorio con las siguientes temáticas:

- Conexión y medición de variables físicas con sensores y Arduino UNO.
- Conexión de Arduino UNO a una red local por medio de shield Ethernet.
- Instalación de sistema operativo Raspbian en Raspberry Pi 3 modelo B con memoria de 32 Gb.
- Instalación de broker Mosquitto en Rasbperry Pi 3.
- Instalación de herramienta Node-RED en Raspberry Pi.
- Comunicación con protocolo MQTT: broker Mosquitto en Raspberry Pi y Arduino UNO como publicador.
- Comunicación con protocolo MQTT: broker Mosquitto en Raspberry Pi y aplicación MyMQTT en smartphone.
- Uso de Node-RED para armado de plataforma online.

Se utilizó en el ámbito de Facultad de Ciencias Físico, Matemáticas y Naturales (ciudad de San Luis) y la Facultad de Ingeniería y Ciencias Agropecuarias (ciudad de Villa Mercedes) de la Universidad Nacional de San Luis en las siguientes actividades:

- Dos conferencias dirigidas a alumnos de las carreras de ingeniería electrónica, tecnicatura en electrónica y afines.
- Dictado del curso Sistemas Distribuidos del Posgrado Maestría en Sistemas Embebidos.
- Talleres para alumnos ingeniería electrónica, tecnicatura en electrónica y afines.
- Eventos de promoción de carreras científicas.
- Materias Procesadores II e Interfaces de las carreras Ingeniería Electrónica con Orientación en Sistemas Digitales, Profesorado en Tecnología Electrónica y Tecnicatura Universitaria en Electrónica.

En el Departamento de Electrónica se tienen 13 docentes como potenciales usuarios, teniendo en cuenta que el dispositivo puede adaptarse a los contenidos de sus asignaturas y proyectos, de los cuales 8 ya lo incluyeron en sus prácticas, teniendo así un 61,53% de respuesta positiva, ver Fig. 4.

Gracias al diseño portátil del prototipo es sencillo su transporte, se puede trasladar a diferentes aulas o edificios sin mayores inconvenientes. El tiempo invertido en la confección de consignas, trabajos prácticos y material de estudio se redujo de manera notable. El prototipo demostró ser versátil frente a diferentes requerimientos y exigencias.

Fig. 4. Docentes usuarios y no interesados.

Con respecto a la utilización frente a alumnos se citará la experiencia en la materia Interfaces. En esta asignatura se realizan cuestionarios previos a cada practica de laboratorio con dos fines:

- Evaluar los conocimientos conceptuales del alumno.
- Determinar si es posible el manejo del hardware por parte del alumno (por motivos de seguridad y cuidado del material de trabajo).

Teniendo en cuenta esto, se realizó un cuestionario con diez preguntas, relacionadas con el manejo de las placas de hardware libre, utilización del protocolo MQTT y la herramienta Node-RED. Es importante destacar que el material de estudio se puso a disposición de los alumnos una semana antes de la práctica de laboratorio. Se contó con un total de 23 alumnos, de los cuales 16 obtuvieron una calificación satisfactoria, representando un 69,56% del total. Para realizar la evaluación posterior al uso del dispositivo, se confeccionó otro cuestionario, valorando los mismos conceptos, pero con diferentes preguntas. Esta vez los resultados satisfactorios obtenidos fueron de 20 alumnos sobre el total de 23, obteniendo así un 86,95%. Se puede visualizar una comparativa de los resultados en las dos instancias en la Fig. 5.

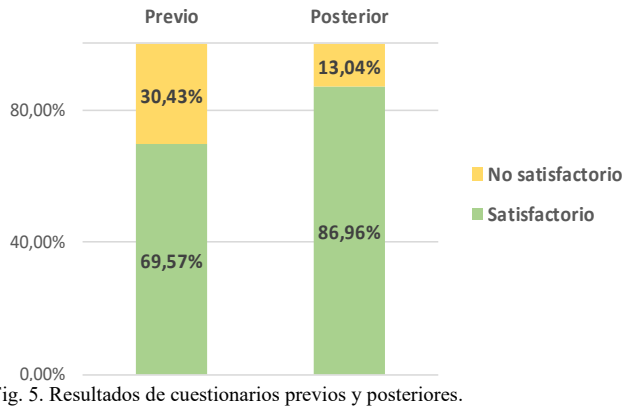

Fig. 5. Resultados de cuestionarios previos y posteriores.

Se comprobó que el uso del prototipo facilita el dictado de la clase debido al interés mostrado por parte de los alumnos al tener un dispositivo armado, con una interfaz amena, con conexión a la red global, entre otras.

Promueve el interés por la temática tanto dentro de la actividad áulica como fuera de ella, haciendo que el alumno pueda desarrollar criterio de búsqueda de información, técnicas de programación y elección de *hardware* y *software* para proyectos.

B. Calibración del Prototipo

Para realizar el análisis de los resultados obtenidos por el prototipo, se utilizaron instrumentos provistos por el Laboratorio de Electrónica, Investigación y Servicios (LEIS) de la Universidad Nacional de San Luis. Todas las pruebas y mediciones se realizaron en el mencionado laboratorio, bajo temperatura ambiente de 25ºC. Los instrumentos utilizados para efectuar las mediciones y calibraciones son: Osciloscopio SIGLENT SDS 204, Multímetro Digital de Precisión HP 34401A (Clase I), fuente de alimentación C.C. de precisión SIGLENT SPD3303S, termómetro digital SCHWYZ SC134, luxómetro digital Protomax modelo MS6610 y estación meteorológica Luft modelo Central Luft.

Los resultados obtenidos de contrastar las mediciones del prototipo con los instrumentos tomados como patrones, se resumen en la Tabla I.

TABLA I
ESPECIFICACIONES FINALES OBTENIDAS

Magnitud	Sensor	Error admitido	Error medido
Temperatura	DHT22	±2°C	1.5°C
Humedad	DHT22V	±5%	4%
Presión Atmosférica	BMP280	5hPa	3hPa
Nivel de iluminación	BH1750	10lux	9lux

En el caso de temperatura, ver Fig. 6, la diferencia promedio obtenida es de 1,22°C, siendo la mínima 0,8°C y la máxima de 1,5°C. Con esto se cumple el objetivo de tener ±2°C de exactitud.

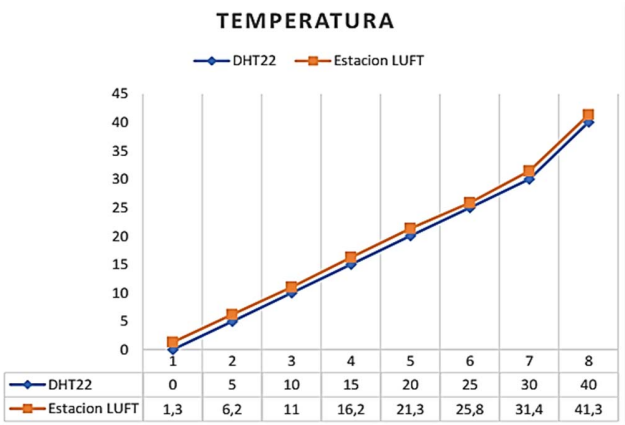

Fig. 6. Comparación de medidas de temperatura.

La diferencia promedio obtenida en el caso de humedad, ver Fig. 7, es de 3,16%, siendo la mínima 2% y la máxima de 4%. Con esto se cumple el objetivo de tener ±5% de exactitud.

La medida de presión atmosférica promedio obtenida es de 1hPa, ver Fig. 8, siendo el mínimo 1hPa y el máximo 3hPa, lográndose la meta de estar dentro de los 5hPa de exactitud.

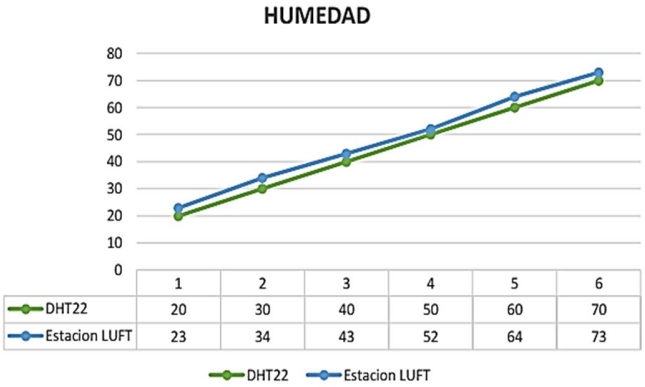

Fig. 7. Comparación de medidas de humedad relativa.

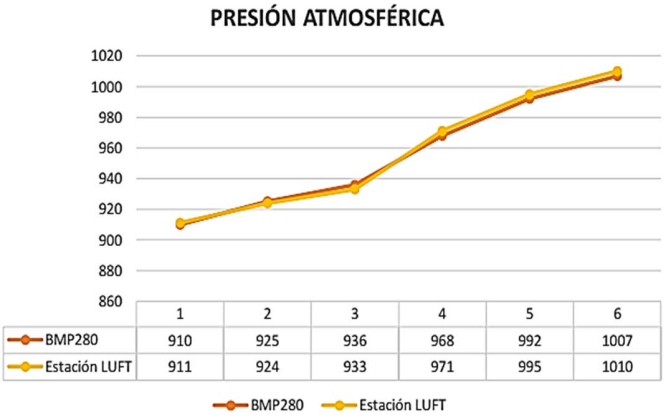

Fig. 8. Comparación de medidas de presión atmosférica.

Para la medición de iluminación, ver Fig. 9, la diferencia promedio obtenida es de 6,2 lux, siendo la mínima 4 lux y la máxima de 9 lux. Con esto se ha cumplido con el objetivo de tener ±10 lux de exactitud.

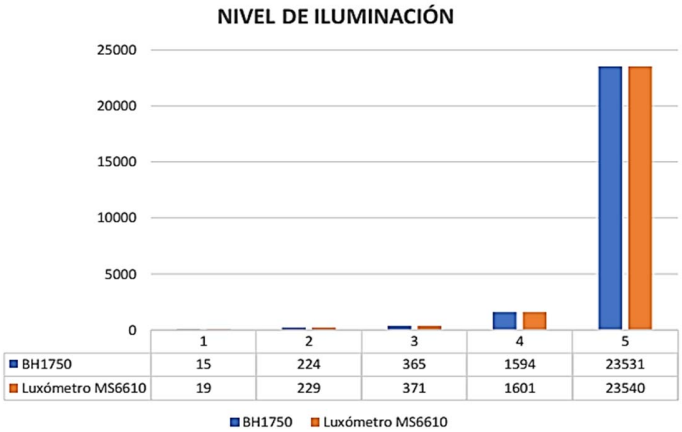

Fig. 9. Comparación de medidas de Nivel de iluminación.

VI. CONCLUSIONES

El interés de los alumnos por incluir en otras prácticas lo aprendido con el protocolo MQTT y la herramienta Node-RED, las preguntas durante y después de las actividades, la predisposición de otros docentes a utilizar el prototipo, fortaleció la idea que la propuesta aquí planteada cumplió con las expectativas iniciales.

Su utilización en la Universidad Nacional de San Luis, para diversas actividades docentes, verificó que el prototipo se adapta al ámbito educativo universitario.

Luego de observar que varios alumnos, dentro de sus posibilidades, replicaron este dispositivo, se deduce que el costo era adecuado. Teniendo en cuenta los resultados obtenidos luego de la utilización del dispositivo desarrollado, se concluye que cumplió satisfactoriamente con los requerimientos.

REFERENCIAS

[1] "*Internet of Things – From Research and Innovation to Market Deployment*", River Publishers Series in Comunications, River Publishers, Aalborg, Dinamarca, 2014, pp. 107-114. [Online]. Disponible: https://www.riverpublishers.com/pdf/ebook/RP_E9788793102958.pdf

[2] OASIS, "*MQTT Versión 5.0 OASIS Standard*" Disponible: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html, 2019.

[3] *"RFC 7252: The Constrained Application Protocol (CoAP)"* Disponible: https://tools.ietf.org/html/rfc7252, 2018.

[4] Node-Red: About. [Online]. Disponible: https://nodered.org/about/

[5] Sofia2: Qué es Sofia2. [Online]. Disponible: https://sofia2.readthedocs.io/en/latest/definicion.html

[6] T. Moraes, B. Nogueira, V. Lira y E. Tavares. Bari. Italia. 2019. "*Performance Comparison of IoT Communication Protocols*". Presentado en: IEEE International Conference on Systems, Man and Cybernetics (SMC).

[7] A. Andrada "Supervisión de Estación Meteorológica Remota por IoT con Tecnología M2M", Trabajo Final de Carrera de Grado, Ingeniería Electrónica OSD, Departamento de Electrónica, Universidad Nacional de San Luis, 2018. https://drive.google.com/open?id=1fHuj_8LDgP18meKK0iZn9iU0rH0VBrXw.

[8] S. Chanthakit, C. Rattanapoka "MQTT Based Air Quality Monitoring System using NodeMCU and Node-RED", Seventh ICT International Student Project Conference (ICT-ISPC), 2018.

[9] R. K. Kodali, A. Anjum "IoT Based HOME AUTOMATION Using Node-RED" Department of E.C.E, National Institute of Technology, Warangal WARANGAL. Second International Conference on Green Computing and Internet of Things (ICGCIoT) 2018.

[10] A. Rajalakshmi H. Shahnasser, "Internet of Things using Node-Red and Alexa", School of Engineering San Francisco State University San Francisco, California. 17th International Symposium on Communications and Information Technologies (ISCIT) 2017.

[11] M. Kusriyanto, A. Anggara Putra "Weather Station Design Using IoT Platform Based On Arduino Mega", International Symposium on Electronics and Smart Devices (ISESD) 2018.

[12] L. Franucci, Agosto, 2012. Buenos Aires, Argentina, "Los sistemas reactivos y la programación dirigida por eventos", Tutorial, Aula 201, Presentado en: "Simposio Argentino de Sistemas Embebidos".

[13] S. Quincozes, E. Tubino, and J. Kazienko "MQTT Protocol: Fundamentals, Tools and Future Directions", IEEE Latin America Transactions, vol. 17, nro. 9, Septiembre, 2019.

[14] Eclipse. Mosquitto an open source mqtt v3.1/v3.1.1 broker. [Online]. Disponible: http://mosquitto.org.

Astri Edith Andrada Tivani. Nacida en Córdoba, Argentina el 21 de Julio de 1989. Ingeniera Electrónica con Orientación en Sistemas Digitales, egresada de la Universidad Nacional de San Luis en 2018. Sus principales líneas de investigación son: Protocolos de Internet de las Cosas, Sistemas embebidos, Sistemas en tiempo Real, Hardware Dinámicamente Reconfigurable, Sistemas Distribuidos. Comenzó su actividad docente en la Universidad Nacional de San Luis en 2013. Actualmente Auxiliar Docente de las Cátedras de Procesadores II e Interfaces, en el Departamento de Electrónica de la Facultad de Ciencias Físico Matemáticas y Naturales, de la Universidad Nacional de San Luis, San Luis, Argentina.

Roberto Martín Murdocca nacido en Buenos Aires, Argentina el 24 de Septiembre de 1974. Ingeniero en Electrónica con Orientación en Sistemas Digitales egresado de la Universidad Nacional de San Luis en 2011.
Áreas de interés: sistemas embebidos, sistemas en tiempo real, RTOS, Linux embebido, adquisición de datos y sistemas distribuidos. Se unió a la Universidad Nacional de San Luis en 1998. Actualmente Profesor de las Cátedras Interfaces y Procesadores II, miembro del Laboratorio de Electrónica, Investigación y Servicios de la Universidad.

Carlos Federico Sosa Paez nacido en San Luis, Argentina. Ingeniero en Electricista de la Universidad Nacional de Córdoba en 1986. Profesor de la Catedra de Diseño de Sistemas Digitales, miembro del Laboratorio de Electrónica, Investigación y Servicios de la Universidad. Área de interés: diseño e implementación de arquitecturas risc en dispositivos lógicos programables de última generación. Es autor y co-autor de varias publicaciones científicas.

Julio Daniel Dondo Gazzano Completó sus estudios de Ingeniería Electricista-Electrónica en la Universidad Nacional de San Luis, en 1996. Recibió el título de Magister en Ingeniería de Software por la misma Universidad en 2007 y el título de Doctor en Ingeniería de Computadores por la Universidad de Castilla-La Mancha en 2010. Sus principales lineas de investigación son: Diseño de SoC basados en FPGAs, Hardware Dinámicamente Reconfigurable, Sistemas Distribuidos Heterogéneos. Actualmente trabaja como Profesor Asociado y es Director de la Carrera de Máster en Sistemas Embebidos, y de la Especialización en Sistemas Embebidos, en el Departamento de Electrónica de la Facultad de Ciencias Físico-Matemáticas y Naturales, de la Universidad Nacional de San Luis, San Luis, Argentina.