Internet de las Cosas y Sistemas Embebidos para monitorear áreas de cultivo.

Peña Koo Jimmy Josué[1]*, Chan May Orlando Adrián[1], Espinoza Galicia Carlos Arturo[2], Gómez López Williams[2], Espinosa Pacho José Ildefonso[1]

[1]*División de Ingeniería en Sistemas Computacionales, Instituto Tecnológico Superior del Sur del Estado de Yucatán, Oxkutzcab, Yucatán, México, * correspondencia: jimjpk@itsyucatan.edu.mx*

[2]*División de Ingeniería en Sistemas Computacionales, Instituto Tecnológico Superior de Huichapan, El Saucillo, Huichapan, Hidalgo, México, * correspondencia: cespinoza@iteshu.edu.mx*

***Resumen*—** Este documento presenta la implementación de una interconexión entre dispositivos de Internet de las Cosas (IoT), mediante la aplicación del protocolo Message Queue Telemetry Transport (MQTT). El objetivo general es el diseño de un sistema de hardware y su configuración para ejecución correcta que permita la recolección de datos tales como temperatura, humedad, entre otros. El sistema consiste en la conexión de dos placas NodeMCU en sus variantes Amica y Lolin, ambas con el SOC ESP8266, comunicados inalámbricamente con una computadora de bolsillo Raspberry Pi 3B+ a través del estándar MQTT que se basa en la modalidad de publicar/suscribir para intercambio de los datos. Lo anterior, sirvió para el envío/recepción de datos en formato CSV y JSON, entre los dispositivos interconectados por tecnologías en la nube entre las que se encuentran MLAB y Azure.

Palabras clave — IoT, NodeMCU, MQTT, JSON, Raspberry Pi

I. INTRODUCCIÓN

Según la FAO (Organización de las Naciones Unidas para la Alimentación y la Agricultura), entre 2005 y 2015 los desastres naturales costaron 96,000 millones de dólares en daños a la producción agrícola y ganadera. Los desastres meteorológicos como las tormentas o las temperaturas extremas causaron pérdidas por valor de 26,500 millones de dólares y los desastres biológicos, como las plagas e infestaciones, hicieron perder 9,500 millones de dólares en cosechas [1].

Considerando lo anterior, en el paradigma IoT existen diversas áreas de interés, una de ellas enfocada al monitoreo de terrenos para la obtención de datos ambientales como la temperatura y humedad ambiental, humedad del terreno o cantidad de lluvia. De esta manera, se colabora en el desarrollo de aplicaciones para solventar los daños causados al medio ambiente en materia de agricultura, por tal motivo, es de interés que se puedan observar mediciones en tiempo real de lo que ocurre en los terrenos de cultivo por medio de sensores interconectados a plataformas en la nube para el almacenamiento y posterior análisis de los diferentes parámetros relacionados con condiciones meteorológicas

II. MATERIALES Y MÉTODOS

2.1. Materiales

Los materiales utilizados para el desarrollo de este sistema son en general: (1) sensores los cuales están conectados a una (2) unidad de recolección de datos el cuál envía la información de cada uno de los sensores a un (3) servidor de almacenamiento el cual concentra la información de cada unidad de recolección y envía estos datos a un servicio web Azure el cual pueda ser consultado en cualquier parte del mundo mediante el Internet.

2.1.1. Sensores

Los sensores utilizados para el monitoreo son los siguientes:

- Sensor DHT 11. Este sensor proporciona el porcentaje de humedad y temperatura del ambiente. Estos sensores constan de dos partes, un sensor capacitivo y un termisor [2]. Además, este dispositivo cuenta con un conversor analógico-digital el cual proporciona una salida digital el cual puede ser leído por cualquier microcontrolador [3].

- Sensor MPL3115A2. Este sensor fabricado por la empresa Sparkfun, mide la presión barométrica de alta precisión y de bajo costo y potencia. Tiene un margen de error de ± 0.05 kPa y cuenta con un conversor análogo-digital de 24 bits de resolución cuya salida se comunica a través del protocolo de comunicación I2C [4].

- Sensor FC-28. Este sensor mide la cantidad de humedad del suelo y materiales similares. Consta de dos almohadillas expuestas actuando juntas como una resistencia variable, cuanta más agua haya en el suelo, mejor será la conductividad entre las almohadillas y por tanto brindará una menor resistencia proporcionando una salida más alta [5].

- Sensor FC-37. Este sensor es el encargado de detectar en qué momento está lloviendo. Se hace uso de los fundamentos descritos en el sensor

FC-28, cuyo ambiente en vez de tierra es el aire [6].

2.1.2. NodeMCU ESP8266

Este módulo, consiste en un kit de desarrollo para el Internet de las Cosas (IoT) similar a Arduino, sin embargo la gran ventaja de este dispositivo respecto al Arduino, es contar con Wifi.

Este dispositivo cuenta con las siguientes características [7]:

- Microcontrolador: Ninguno.
- Procesador: Tensilica Xtensa LX106
- Fuente de poder: 5 V desde USB o 3.3 V de VIN
- Voltaje de Entrada/Salida: 3.3 V
- Memoria RAM: 64 KB por instrucción Interna y 96KB de memoria para datos.
- Memoria ROM: QSPI flash 512 KB a 4 MB.
- Interfaz serie: SPI, I2C, I2S y UART
- Puertos discretos: 10 pines digitales de entrada y salida (pueden ser usados para PWM, I2C y 1-wire) y 1 pin de entrada analógica de 10 bits.
- Herramienta de programación: NodeMCU Firmware y Arduino IDE
- Lenguajes de Programación: LUA y C/C++.

2.1.3. Raspberry Pi.

Raspberry Pi es un dispositivo embebido el cual es una iniciativa británica el cual consiste en crear un microordenador económico y flexible para ser usado para diferentes usos. Su diseño incluye un Broadcom BCM2835 con un procesador ARM1176JZFS a 700 MHz con una unidad de coma flotante, 1GB de memoria RAM y procesador gráfico Videocore IV. La tarjeta carga desde una memoria MicroSD el SO y cuenta con dos conectores USB 2.0, conector Ethernet, salida de video digital HDMI [8].

2.1.4. MongoDB

MongoDB es una base de datos documental de código abierto y líder en bases de datos NoSQL.

MongoDB está escrito en C++. Ofrece alta disponibilidad, escalabilidad y particionamiento a costa de consistencia y soporte transaccional. En términos prácticos, esto significa que, en lugar de tablas y filas, MongoDB utiliza documentos para hacerla flexible, escalable y rápida. Guarda estructuras de datos en documentos similares a JSON con un esquema dinámico (MongoDB utiliza una especificación llamada BSON para almacenar sus documentos), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

Los documentos BSON sirven para mantener una lista ordenada de elementos, estos elementos tienen tres componentes: 1) un nombre de campo, 2) un tipo de datos y 3) un valor. Estos documentos pueden tener esquemas diferentes, lo que significa que el esquema puede cambiar a medida que evoluciona la aplicación".

2.1.5. Valores separados por comas (CSV)

Un archivo CSV (Comma Separated Values, Valores Separados por Coma) contiene datos del tipo de los que se encuentran en bases de datos en formato de texto simple. Como el nombre indica cada valor está separado del siguiente por una coma. Formato. Este RFC documenta el formato de los valores separados por comas Estos archivos pueden ser creados, visionados y editados con programas como Microsoft Excel o incluso un simple editor de textos.

2.1.6. JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

2.1.7. Microsoft Azure

Microsoft Azure es conjunto en constante expansión de servicios en la nube para ayudar a las organizaciones a satisfacer sus necesidades comerciales. Otorga la libertad de crear, administrar e implementar aplicaciones en una red mundial con sus herramientas y marcos favoritos.

Azure es una nube pública de pago por uso que permite compilar, implementar y administrar rápidamente aplicaciones en una red global de datacenters (centros de datos) de Microsoft.

2.1.8. Asp.Net Core

ASP.NET Core es un marco multiplataforma de código abierto y de alto rendimiento que tiene como finalidad compilar modernas aplicaciones

conectadas a Internet y basadas en la nube. Con ASP.NET Core puede hacer lo siguiente:

- Compilar servicios y aplicaciones web, aplicaciones de IoT y back-ends móviles.
- Usar sus herramientas de desarrollo favoritas en Windows, macOS y Linux.
- Efectuar implementaciones locales y en la nube.
- Ejecutarlo en .NET Core o en .NET Framework.

ASP.NET Core ofrece las siguientes ventajas:

- Un caso unificado para crear API web y una interfaz de usuario web.
- Integración de marcos del lado cliente modernos y flujos de trabajo de desarrollo.
- Un sistema de configuración basado en el entorno y preparado para la nube.
- Inserción de dependencias integrada.
- Una canalización de solicitudes HTTP ligera, modular y de alto rendimiento.
- Capacidad de hospedarse en IIS, Nginx, Apache, Docker o de autohospedarse en su propio proceso.
- Control de versiones de aplicaciones en paralelo con .NET Core como destino.
- Herramientas que simplifican el desarrollo web moderno.
- Capacidad para compilarse y ejecutarse en Windows, macOS y Linux.
- De código abierto y centrado en la comunidad.

ASP.NET Core se distribuye en su totalidad como paquetes NuGet. El uso de paquetes NuGet permite optimizar la aplicación para incluir únicamente las dependencias necesarias. De hecho, las aplicaciones ASP.NET Core 2.x que tienen .NET Core como destino solo requieren un paquete NuGet único. Entre las ventajas de una menor superficie de aplicación se incluyen una mayor seguridad, un mantenimiento reducido y un rendimiento mejorado.

2.1.9. MQTT

Message Queue Telemetry Transport (MQTT), es un protocolo de mensajería de publicación/suscripción, simple y ligero, diseñado para dispositivos con restricciones y redes de bajo ancho de banda, alta latencia o poco confiables, por lo tanto, es ideal para proyectos de IoT o Máquina a Máquina (M2M), por permitir la comunicación de dispositivos que operan con baterías de bajo consumo de energía. Además, es adecuado para la comunicación con diversos sensores inteligentes, así como con microcontroladores y computadoras de bajo costo [9].

El protocolo opera en un modelo basado en cliente/servidor, donde el servidor central, conocido como bróker, recibe mensajes de los clientes, los cuales esencialmente son todos los nodos incluidos en la comunicación. Ofrece privacidad, autenticación y seguridad sacándole ventaja a la utilización de User Datagram Protocol (UDP) [10].

Como se menciona en [11], esos mensajes pueden ser tópicos de publicaciones o suscripciones. Esta estructura permite una comunicación muchos a muchos y desacoplamientos de productor y consumidor (ver figura 1).

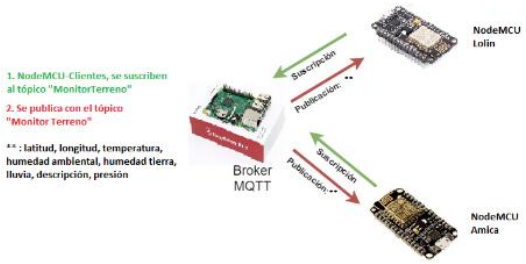

Figura 1. Arquitectura MQTT publicación/suscripción

La arquitectura, funciona del siguiente modo: a pesar que cualquier dispositivo puede publicar y suscribirse, quizá incluya un conjunto de sensores que periódicamente publiquen sus mediciones para un tópico específico, de esta manera, cualquier dispositivo registrado y que esté interesado como suscriptor de un tópico específico, puede recibir un mensaje del bróker cada vez que el tópico se actualiza. Los tópicos de MQTT son jerárquicos (ejemplo, escuela/aula/temperatura). El puerto 1883 se reserva para este protocolo.

Para este caso de estudio, se trabajó con dos módulos NodeMCU en sus versiones de Amica y Lolin así como una computadora de bolsillo Raspberry Pi 3.

El transporte de mensajes es independiente del contenido de la carga útil, donde existen tres niveles o calidades de servicio (QoS) para la entrega de éstos: a lo sumo una vez, al menos una vez y exactamente una vez [12].

En el proyecto se utilizó el nivel “a lo sumo una vez”, con datos de sensores ambientales, porque no importa si una lectura individual es perdida ya que la próxima será publicada poco después, cada dos minutos.

Se decidió emplear este protocolo porque como afirman [13], la ventaja de MQTT es la creación de un protocolo abierto, que permite la conexión de cualquier tipo de dispositivo que admita el protocolo TCP/IP, de este modo, cuando un cliente envía un mensaje a un tema y

otro cliente se encuentra suscrito al tema, recibe el mensaje. Para el proyecto se empleó el tópico "MonitorTerreno".

2.1.10. HTTP-Rest

Representational State Transfer (REST), es un estilo de arquitectura de software para sistemas distribuidos como la Web. En realidad, REST se refiere estrictamente a una colección de principios para el diseño de arquitecturas en red. Estos principios resumen cómo son definidos los recursos. El término frecuentemente es utilizado en el sentido de describir a cualquier interfaz que transmite datos específicos de un domino sobre HTTP.

El estilo REST enfatiza que las interacciones entre clientes y servicios se mejoran al tener un número limitado de operaciones o verbos y una interfaz estándar entre los componentes proporcionada por el protocolo HTTP. La flexibilidad es posible asignando a cada recurso su propio URL, esencialmente dándole un número ilimitado de sustantivos. Al dar un significado exacto a los verbos HTTP de GET, POST, PUT y DELETE, REST evita la ambigüedad. Por ejemplo, GET no puede crear algún efecto secundario, sino que simplemente devuelve una representación de recursos [14].
El concepto de recurso es definido por [15] como cualquier elemento de interés. Por ejemplo, Boeing Aircraft Corp, puede definir un recurso 747. Los clientes pueden acceder a ese recurso con esta URL: http://www.boeing.com/aircraft/747.
Posteriormente, se devuelve una representación del recurso, por ejemplo, Boeing747.html, donde éste coloca la aplicación cliente en un estado. El resultado del cliente que atraviesa un hipervínculo en Boeing747.html accede a otro recurso. La nueva representación coloca la aplicación del cliente en otro estado y así sucesivamente en cada representación de recursos.

En el trabajo de disertación original, acerca de transferencia de estado representacional [16], explica que si bien REST no es un estándar, sí utiliza estándares como HTTP y URL, además de representaciones de recursos como XML / HTML / GIF / JPEG, entre otros, e incluso algunos tipos MIME como text/xml, text/html, image/gif, image/jpeg. También, señala que la Web funciona mejor cuando usa REST.

2.2. Metodología

2.2.1. De la investigación

De acuerdo a la clasificación de Hernández [17], esta investigación tiene un diseño experimental de clase pre-experimental, porque se administró un tratamiento a dos grupos y después se aplicó una medición de cinco variables (Temperatura, Humedad Ambiental, Humedad de la Tierra, Lluvia y Presión Barométrica) para observar el nivel de las áreas de cultivo en éstas. Adicionalmente, los grupos de estudio no se asignaron al azar ni se emparejaron, se aplicó a dos terrenos de cultivo disponibles para pruebas y experimentación.

El diseño de la investigación está representado con la siguiente simbología:

$$GXO$$

donde:

G: Representa las áreas de cultivo empleadas para las pruebas y experimentación durante la recolección de información.

X: Tratamiento, consiste en el sensado de variables involucradas en el crecimiento de los cultivos.

O: Medición de variables involucradas en el cultivo.

2.2.2. Del experimento

El diseño del experimento se basó en una adaptación de la metodología Modelo en V, dado que el Modelo en V surge para proyectos de desarrollo de software, como un modelo robusto para proyectos pequeños, con equipos de hasta cinco personas [18].

La estructura metodológica adaptada se presenta en cuatro niveles (ver figura 2): el primer nivel es orientado al cliente final, constituido por el inicio y fin del proyecto como extremos del ciclo, se compone de la definición de requerimientos y pruebas operacionales o de campo; el segundo nivel, se enfoca a las características funcionales del sistema, integrado por el diseño (hardware y software) y prueba de integración; la tercera fase, define la arquitectura a nivel módulo, integrada por la configuración, codificación y pruebas unitarias de cada módulo que integra el sistema; y por último, el nivel cuatro, integración, en el cual se une el desarrollo de cada módulo de hardware y software del sistema.

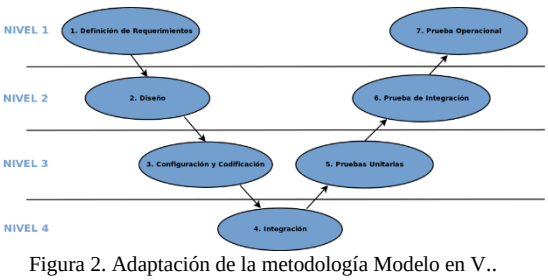

Figura 2. Adaptación de la metodología Modelo en V..

III.RESULTADOS Y DISCUSIÓN

3.1. Resultados

A partir de la aplicación de la metodología modelo en V, se obtuvieron los siguientes resultados en cada una de las etapas.

3.1.1. Definición de requerimientos

Por medio del análisis de trabajos relacionados y entrevistas, se definieron las variables a observar, involucradas para el cultivo en los terrenos experimentales, resultando: Temperatura Ambiental, Humedad Ambiental, Humedad en la Tierra, Lluvia y Presión Barométrica.

Para la monitorización, se requiere visualizar información de las variables sensadas en plataforma Web y por medio de una aplicación móvil, por lo que se identificaron los requerimientos del desarrollo de ambas plataformas.

3.1.2. Diseño

Resultado de la identificación de las variables a medir, se eligieron los módulos para monitorear las áreas de cultivo, compatibles con la placa de IoT NodeMCU, resultando de la siguiente manera:

Tabla 1. Relación de módulos empleados

Variable	Módulo	Tipo
Temperatura Ambiental	DHT11	Digital
Humedad Ambiental	DHT11	Digital
Humedad en la Tierra	FC28	Analógico / Digital
Lluvia	FC37	Analógico / Digital
Presión Barométrica	MPL311	Digital
Hora y Fecha de Sensado	DS3231	Digital

En la parte de software, se realizó el diseño de las interfaces de software a desarrollar para las plataformas móviles y web, basadas en el Lenguaje Unificado de Modelos.

3.1.3. Configuración y codificación

A partir de la identificación de los módulos, se conectaron acorde a los pines disponibles del NodeMCU e instalaron librerías para el desarrollo de la aplicación con el entorno Arduino IDE (ver Tabla 2).

Se empleó una computadora de bolsillo Raspberry Pi, como broker para el servicio de publicación de tópicos empleando el protocolo MQTT. El servidor recibe y almacena en formato CSV los datos sensados por los módulos conectados a la interfaz de IoT NodeMCU. La interfaz NodeMCU se suscribe por medio del mismo tópico declarado en el servidor, para el envío de datos al broker por medio del protocolo MQTT.

Tabla 2. Pines del NodeMCU usados para los módulos del prototipo

Módulo	Pin del NodeMCU	Controlador
DHT11	D4	Pubsubclient.h, DHT11.h
FC28	A0	Incluído en el IDE
FC37	D5	Incluído en el IDE
MPL311	D1, D2	SparkFunMPL3115A2.h, Wire.h
DS3231	D2	Wire.h, RTClib.h

Se desarrolló un interfaz web y móvil para visualizar el comportamiento de los datos sensados, empleando programación por capas con el entorno de programación .NET, la cual comparte código entre las aplicaciones web y móviles mediante bibliotecas .NET Standar, la información es almacenada en una base de datos NoSQL, MongoDB, que se conecta mediante un driver provisto para .NET; la parte web fue desarrollada usando ASP.Net core, la cual contiene una API REST Full, la parte móvil se desarrolló utilizando Xamarin Forms para permitir compilar la aplicación para Android y iOS.

También se desarrolló una aplicación en el entorno Python, como puente de comunicación entre el broker y el repositorio de base de datos Mongo. La aplicación extrae la información en formato CSV, la convierte a JSON y posterior la publica por medio de una solicitud POST.

3.1.4. Integración

En esta fase se integró el desarrollo de software con el prototipo de hardware de acuerdo con la figura 3, dando paso a diferentes pruebas para la implementación de la experimentación.

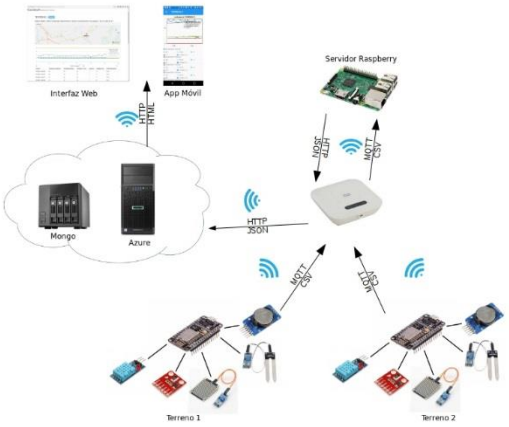

Figura 3. Modelo del Proyecto Monitor Terreno.

3.1.5. Pruebas Unitarias

Estas pruebas se realizaron para identificar la funcionalidad de la lectura de cada sensor, el envío al broker en formato CSV, la transformación a formato JSON y la visualización en las plataformas Web y Móvil.
Para esta etapa se realizaron programas independientes en Arduino IDE, se observaban las cotas que el sensor devolvía y se hicieron ajustes por medio de ecuaciones lineales para almacenar la información.

3.1.6. Prueba de Integración

Para la prueba de integración, se empleó el prototipo implementado en una placa universal, incluyendo los sensores DHT11, FC28, FC37 y MPL311, y el módulo de reloj DS3231 (ver figura 4), con el propósito de observar la funcionalidad del envío de todos los atributos por medio del protocolo MQTT.

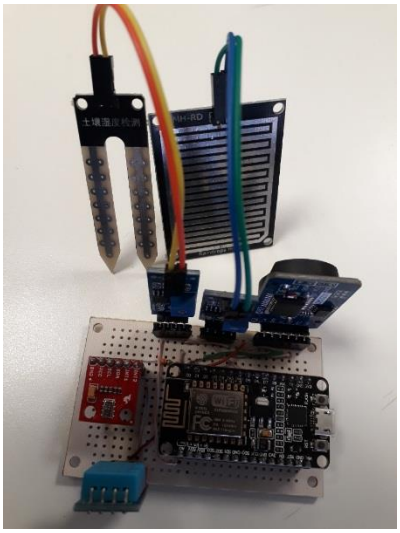

Figura 4. Prototipo implementado en placa universal.

3.1.7. Prueba Operacional

Para esta última etapa se diseñó el PCB (ver figura 5) para implementar el prototipo en dos áreas de cultivo disponibles para la experimentación.

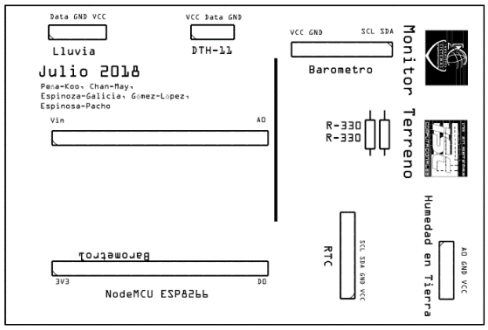

Figura 5. PCB del prototipo Monitor Terreno.

Con el propósito de identificar los valores enviados por cada uno de los terrenos, se añadió la descripción del terreno, su latitud y longitud, cuyos valores fueron tomados empleando el WiFi de la placa IoT NodeMCU.

3.2. Discusión

IoT es una tecnología que ha tenido un fuerte impulso en años recientes y lo seguirá teniendo por la aparición de diferentes plataformas tanto de hardware como de software, apoyadas por la filosofía de open source, lo que lo hace más atractivo para el desarrollo de diversos proyectos. Otro aspecto que lo impulsa, es la facilidad para obtenerlos tanto en distribución como en costos.
Con la implementación e interrelación de módulos, placas, plataformas y, diversos sensores se obtuvieron resultados satisfactorios en cuanto al monitoreo y el envío de información, objetivo principal del trabajo.

De igual manera, con este trabajo es posible afirmar que a partir de una buena conexión inalámbrica WiFi, es posible la transferencia de datos entre los diversos elementos que conforman la arquitectura IoT, sin tener dificultades, tanto a nivel local como en bases de datos en la nube. Se obtuvieron diversos valores durante ocho días ininterrumpidos, por lo que la tecnología es estable para la medición de dichas variables y su posible aplicación en diversas áreas que así lo requieran. Los registros de datos generados proporcionan una amplia gama de conclusiones y opciones para implementarse en sistemas propios o desarrollados a nivel de investigación. También, a partir del trabajo se propone una metodología acorde a las necesidades de implementación de proyectos de IoT.

Aunado a lo anterior, se encuentra la posibilidad de aplicarlo en proyectos concretos, sobre todo aquellos relacionados con la producción dentro de áreas relacionadas con la agricultura. Así, la posibilidad de generar nuevos trabajos de investigación está latente, incluso, a mediano plazo se pretende aplicarlo en el control de plantas de stevia, incluyendo sensores de ph, además de los ya mencionados. Otros beneficios para los usuarios son el acceso en tiempo real de las condiciones del área o terreno, la reducción en el uso de recursos a través de la aplicación exacta en los tiempos adecuados, contribuyendo así a una producción más sustentable. A largo plazo, se pretende trabajar en la seguridad del sistema, aplicando mecanismos de encriptación de datos M2M, característica que posee el protocolo de comunicación.

Diversos trabajos presentan similitudes al proponer soluciones para diferentes aplicaciones

como la agricultura, ciudades inteligentes y cuidado de la salud, así como aplicaciones tecnológicas para control y automatización de procesos agrícolas para el monitoreo de variables físicas como la temperatura en lo que se ha denominado agricultura de precisión [19].

Algunos sistemas similares existentes, en el área IoT se mencionan a continuación:
EcoStruxure, Es la arquitectura y plataforma interoperativa abierta, habilitada para IoT y de inicio automático de Schneider Electric, para hogares, edificios, centros de datos, infraestructura e industrias que ofrece innovación en todos los niveles, desde productos conectados a control perimetral, y aplicaciones, análisis y servicios, energía, TI, construcción, maquinaria, plantas y redes [20].

SGreenH-IoT: Plataforma IoT para Agricultura de Precisión. Una plataforma IoT de bajo costo y consumo energético para la monitorización de campos de cultivo e invernaderos, conformada por una arquitectura en capas, un protocolo de comunicación, el diseño de un nodo de bajo costo y consumo de energía y una aplicación web para la visualización de los datos [19].

Propuesta de una arquitectura para Agricultura de Precisión soportada en IoT. La propuesta se basa en la estructura de la arquitectura Lambda, considerando diferentes capas: captura de datos, cuya función es la obtención de variables asociadas a un cultivo. Capa de almacenamiento, cuya función es recopilar la información en tiempo real desde los sensores. Capa de procesamiento, genera predicciones y recomendaciones, la cual es evaluada mediante pruebas de carga con el fin de determinar su capacidad y tiempo de respuesta. Capa de consulta, permite a los usuarios finales visualizar en una interfaz web los datos climáticos y las predicciones. Así, esta arquitectura pretende servir de referencia para la implementación de servicios basados en IoT en el área de la agricultura [22].

Considerando los trabajos anteriores, es posible notar la implementación de la solución con elementos de bajo costo, protocolos de comunicación, y una interfaz de visualización sencilla, que recibe datos del monitoreo del ambiente y transmite toda la información a la nube para su graficación posterior, además, se encuentra en un enfoque emergente en el paradigma de IoT y su aportación en el cuidado medio ambiental.

IV. CONCLUSIONES

Las pruebas unitarias de los diversos sensores empleados en la parte experimental mediante la placa para proyectos IoT, denominado NodeMCU, tanto a nivel de circuitería como de verificación y ejecución de código, fueron totalmente compatibles con las versiones de Arduino. Por consiguiente, es importante hacer mención que existe facilidad en la aplicación de conceptos previos de domótica, para su posible adaptación y actualización dentro del paradigma de IoT, sin necesidad de obtener componentes aislados. La combinación o inclusión de esquemas modulares permite la generación de proyectos integrales, con la inclusión de las librerías correspondientes para su funcionamiento con el chip ESP8266, en sus diversas variantes.

Se realizó la lectura de dos áreas de cultivo cada dos minutos por siete días. De las mediciones se obtuvieron un total de 9,743 registros con información relevante de las condiciones de ambos terrenos, entre los que se encuentran la temperatura ambiental, la cual estuvo entre 23 y 25 oC; humedad ambiental entre 31 y 42 por ciento; humedad en la tierra 18 por ciento; un estado para indicar presencia o ausencia de lluvia, así como para determinar la cantidad de ésta en el área y; los valores de la presión barométrica de 79.12 pascales.

Los datos anteriores, fueron presentados a través de una interfaz en la nube, como se observa en la figura 6, para la visualización del comportamiento de los datos obtenidos. La información se despliega en tiempo real, por consiguiente las principales variables que intervienen en la producción de las áreas de cultivo se observan inmediatamente, y con una interpretación adecuada se convierte en información útil para ayudar en las tomas de decisiones en materia de producción y eficiencia ambiental.

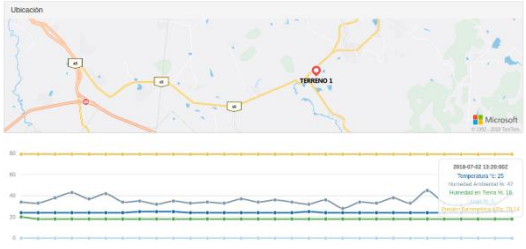

Figura 6. Interfaz de visualización de los sensores.

También, los resultados obtenidos en la experimentación mostraron el funcionamiento eficiente del sistema donde el porcentaje de pérdida de datos en la transmisión fue

prácticamente nula. Esto fue posible a través del modelo de experimentación propuesto, porque se enfoca en aspectos importantes de pruebas, integración e implementación, etapas primordiales en los proyectos actuales de IoT y que no se adaptan en su totalidad con otros modelos de desarrollo propuestos como el modelo V.

Para finalizar, se recomienda a los desarrolladores utilizar módulos integrales para desarrollar proyectos IoT, como es el caso de NodeMCU y Raspberry Pi, por su sencillez y similitudes con otros entornos de programación, de igual manera, aplicarlos en áreas que sean para beneficio del medio ambiente.

REFERENCIAS

[1] FAO (2017). Los desastres causan pérdidas agrícolas millonarias, con la sequía a la cabeza. [Edición electrónica]. Recuperado de http://www.fao.org/3/I8656EN/i8656en.pdf

[2] L. Ada, «DHT, DHT22 and AM2302 Sensors,» adafruit learning system, Vols. %1 de %2-, nº -, pp. 1-13, 2018.

[3] S. Nate, «Sparkfun Start Something,» Sparkfun, [En línea]. Available: https://learn.sparkfun.com/tutorials/mpl3115a2-pressure-sensor-hookup-guide?_ga=2.225175379.814946704.1530818913-1341218842.1530025772. [Último acceso: 05 07 2018].

[4] A. o. Circuits, «Art of Circuits,» [En línea]. Available: https://artofcircuits.com/product/fc-28-soil-moisture-sensor-analog-and-digital-outputs. [Último acceso: 05 07 2018].

[5] S. Pasha, «Thingspeak Based Sensing and Monitoring System for IoT with Matlab Analysis,» International Journal of New Tchnology and Research, vol. 2, pp. 19-23, 2016.

[6] P. S. E. C. G. C. M. R. E. Plaza, «Wireless Development Boards to Connect the World,» Springer International Publishing, pp. 19-27, 2018.

[7] [7] A. S. A. V. M. Valera, «Plataformas de Bajo Coste para la Realización de Trabajos Prácticos de Mecatrónica y Robótica,» ELSEVIER, vol. 11, pp. 363-376, 2014.

[8] J. Castillo, J. Garcés, M. Navas y F. Segovia, «Base de Datos NoSQL: MongoDB vs. Cassandra en operaciones CRUD,» Revista Publicando, pp. 79-107, 2017.

[9] MQTT (2018). What is QMTT? Recuperado de http://mqtt.org/faq

[10] Zambrano V. A., Pérez L. I., Palau S.C., Esteve D. M. (2015). Sistema Distribuido de Detección de Sismos Usando una Red de Sensores Inalámbrica para Alerta Temprana. Revista Iberoamericana de Automática e Informática Industrial RIAI. 12, 260-269.

[11] Prada M. A., Perfecto R. S., Morán A., Fuertes J. J, Domínguez M. (2016). Communication with resource-constrained devices through MQTT for control education. International Federation of Automatic Control (IFAC). Elsevier. 49, 150–155

[12] Banks A., Gupta R. (2014). MQTT Version 3.1.1. [Edición electrónica]. Recuperado de http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/csprd02/mqtt-v3.1.1-csprd02.pdf

[13] Matabuena D., Bellido F.J., Moreno M. A., Gil-de-Castro A., Flores J.M. (2018). Educational platform for communications using the MQTT protocol. Resúmenes Congreso de Tecnología, Aprendizaje y Enseñanza de la Electrónica [Edición electrónica]. Recuperado de http://taee2018.org/wp-content/uploads/2017/07/Libro-de-Res%C3%BAmenes-TAEE-2018.pdf

[14] Brodgen W. (2014). REST versus SOAP. The REST story. Recuperado de http://searchwebservices.techtarget.com/tip/0,289483,sid26_gci1227190,00.html

[15] Costello R. L. (2015). Building Web Services the REST Way. Recuperado de http://www.xfront.com/REST-Web-Services.html

[16] Fielding R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Original Disertation, University of California, Irvine. [Edición electrónica]. Recuperado de https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[17] Hernández, R., Fernández, C. & Baptista, P. (2010). Metodología de la Investigación. México: Mc Graw Hill.

[18] Firesmith, F. (2013). Using V Models for Testing. julio 1, 2018, de Software Engineering Institute, Carnegie Mellon University Sitio web: https://insights.sei.cmu.edu/sei_blog/2013/11/using-v-models-for-testing.html

[19] Guerrero J. A., Estrada G. P., Medina T. M., Rivera G. M., Alcaraz A. J. (2017). SgreenH-IoT: Plataforma IoT para agricultura de precisión. Sistemas, Cibernética e Informática. Vol. 14, pp. 53-58.

[20] E. Schneider. EcoStruxure: IoT-enabled architecture and platform. Disponible en https://www.schneider-electric.com.mx/es/work/campaign/innovation/ [Último acceso: 07 09 2018]

[21] E. Quiroga, S. Jaramillo, W. Campo, G. Chanchí. Propuesta de una Arquitectura para Agricultura de Precisión Soportada en Io. Revista Ibérica de Sistemas e Tecnologias de Informação. Vol 24. pp. 39-56.*Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 1.