INVESTIGACIÓN & DESARROLLO, Vol. **18**, No. **1**: 103 - 116 (2018)
ISSN 2518-4431

SISTEMA DE MONITOREO ENERGÉTICO Y CONTROL DOMÓTICO BASADO EN TECNOLOGÍA "INTERNET DE LAS COSAS"

AN ENERGY MONITORING AND DOMOTIC CONTROL SYSTEM BASED ON "INTERNET OF THINGS" TECHNOLOGY

Erick Escobar Gallardo[1] y Alex Villazón[2]
[1]*Centro de Investigaciones Ópticas y Energías* (CIOE)
[2]*Centro de Investigaciones de Nuevas Tecnologías Informáticas* (CINTI)
Universidad Privada Boliviana
(Recibido el 2 de Mayo 2018, aceptado para publicación el 17 de Junio 2018)

RESUMEN

El "Internet de las Cosas" (Internet of Things - IoT por sus siglas en inglés) es una tecnología en la que pequeños dispositivos electrónicos pueden conectarse a internet, permitiendo el desarrollo de nuevas aplicaciones y servicios. Recientemente, la aplicación de la tecnología IoT a sistemas de eficiencia energética ha despertado interés, sobre todo para el monitoreo de la eficiencia de sistemas en tiempo real. Este artículo presenta el diseño y desarrollo de ChuchusMOTE, un sistema de monitoreo energético y control domótico a través del uso de una red de sensores y actuadores inalámbricos, que utilizan el protocolo de comunicación asíncrono MQTT (Message Queue Telemetry Transport) para el envío de datos y permite su visualización en tiempo real. Se desarrollaron módulos electrónicos implementados sobre la plataforma de desarrollo NodeMCU, una placa hardware inalámbrica que incluye módulos de comunicación compatible con el protocolo MQTT. El sistema ChuchusMOTE fue desplegado para monitorear en tiempo real variables energéticas de paneles solares (voltaje, corriente, potencia y energía generada), el consumo de energía eléctrica y estimar el consumo energético dentro del "Laboratorio de Energías Renovables" de la Universidad Privada Boliviana. Asimismo, ChuchusMOTE controla y automatiza el sistema de iluminación, calefacción solar y extracción de aire de dicho laboratorio.

Palabras Clave: Internet de las Cosas, Monitoreo energético, Domótica.

ABSTRACT

The "Internet of Things" or simply IoT, is a technology where small electronic devices can be connected to the internet, allowing the development of new applications and services. Recently, applying IoT technology to energy efficiency systems has attracted interest, notably for real-time efficiency monitoring. In this article, we present the design and development of ChuchusMOTE, an energy monitoring and domotic control system, based on a network of wireless sensors and actuators communicating through MQTT (Message Queue Telemetry Transport), an asynchronous protocol that is used to send data and enables real-time visualization. We implemented electronic modules based on NodeMCU, a wireless development board that includes communication modules that are compatible with the MQTT protocol. ChuchusMOTE was deployed to monitor in real-time energy variables of solar panels (voltage, current, power and generated energy), the consumption of electrical energy, and estimate the energy consumption of the "Renewable Energy Laboratory" at Universidad Privada Boliviana. Furthermore, ChuchusMOTE controls and automatize the lab's lightning, solar heating and air extraction systems.

Keywords: Internet of Things, Energy monitoring, Domotics.

1. INTRODUCCIÓN

El "Internet de las cosas" o "Internet of Things" (que en el resto del artículo referiremos simplemente como IoT, por sus siglas en inglés), es la tecnología que permite la interconexión de dispositivos físicos, vehículos, edificios y otros elementos integrados con electrónica, software, sensores, actuadores y redes de conectividad que permiten a estos objetos recopilar e intercambiar datos para su posterior análisis [1]. A diferencia del procesamiento y control de la señal digital tradicional que consumen demasiada potencia en los nodos y demasiado ancho de banda en la red al procesar series de tiempo periódicas, la tecnología IoT permite realizar *muestreo por evento* o *muestreo aperiódico* reduciendo así el consumo energético.

Por otro lado, los sistemas de control y monitoreo en tiempo real son importantes en múltiples áreas y aplicaciones, gracias a que otorgan la habilidad de poder monitorear la eficiencia del sistema analizado, además de poder diagnosticar errores y fallas presentes en estos. Estas fallas se ven reflejadas en las múltiples herramientas disponibles en los sistemas de monitoreo en tiempo real, e.g. a través de gráficas de las variables monitoreadas. Dentro de los sistemas de control, el control domótico permite automatizar sistemas eléctricos y electrónicos ejecutando rutinas programadas para así mejorar la eficiencia energética de estos sistemas y proveer facilidad de manejo al usuario final.

DOI: 10.23881/idupbo.018.1-8i

Combinando características de monitoreo y control domótico con la tecnología IoT, se puede obtener un sistema de análisis de eficiencia y control de cargas eléctricas y electrónicas para poder ver así el efecto de incidencia sobre el sistema analizado y el consumo total energético. Debido a las restricciones en potencia y ancho de banda, el IoT permite la computación distribuida de los eventos generados por los sensores. Si bien los procesadores relativamente pequeños de los sensores pueden realizar un procesamiento útil en muchas secuencias de datos, el reconocimiento de eventos de interés, mediante el procesamiento externo y envío de datos utilizando un protocolo ligero, reduce la cantidad de ancho de banda de la red consumida y también reduce el consumo de energía ya que la comunicación inalámbrica que requiere grandes cantidades de energía [2].

En este artículo se describe el diseño y desarrollo de ChuchusMOTE[1] un sistema de monitoreo de variables energéticas y control domótico basado en la tecnología IoT, que consta de nodos electrónicos sensores y actuadores encargados de obtener variables energéticas y controlar remotamente cargas eléctricas y electrónicas. Este sistema sirve para aplicaciones de análisis y mejora de eficiencia de sistemas fotovoltaicos, así como aplicaciones de análisis de consumo energético de un entorno. Utilizando la tecnología IoT se puede conectar sensores con los módulos de control y análisis de datos, a través de mensajes ligeros de manera asíncrona. Asimismo, el sistema permite guardar todos los datos en una base de datos, y visualizar todas las variables controladas en tiempo real a través de aplicación Web.

El resto del artículo está estructurado de la siguiente manera: La Sección 2 presenta la arquitectura de ChuchusMOTE. La Sección 3 describe en detalle el diseño, desarrollo del prototipo incluyendo el software de control y visualización, y una estimación de costo del sistema desarrollado. Los resultados obtenidos se muestran en la Sección 4 y finalmente la Sección 5 concluye el artículo.

2. ARQUITECTURA DEL SISTEMA CHUCHUSMOTE

Para poder separar, de acuerdo a su funcionalidad, los diferentes componentes de un sistema IoT, se considera la taxonomía definida por Gubbi *et al*. [3] donde definen 3 elementos genéricos en una arquitectura IoT:

(1) **Hardware:** Compuesto por sensores, actuadores y hardware de comunicación.
(2) **Middleware:** Capa de software que incluye protocolos de comunicación, almacenamiento y herramientas de cómputo para análisis de datos
(3) **Presentación:** Visualización y herramientas de interpretación que pueden utilizar para diferentes aplicaciones.

La arquitectura del sistema ChuchusMOTE descrita en la Figura 1, sigue una estructura IoT en 3 niveles, como se describe a continuación.

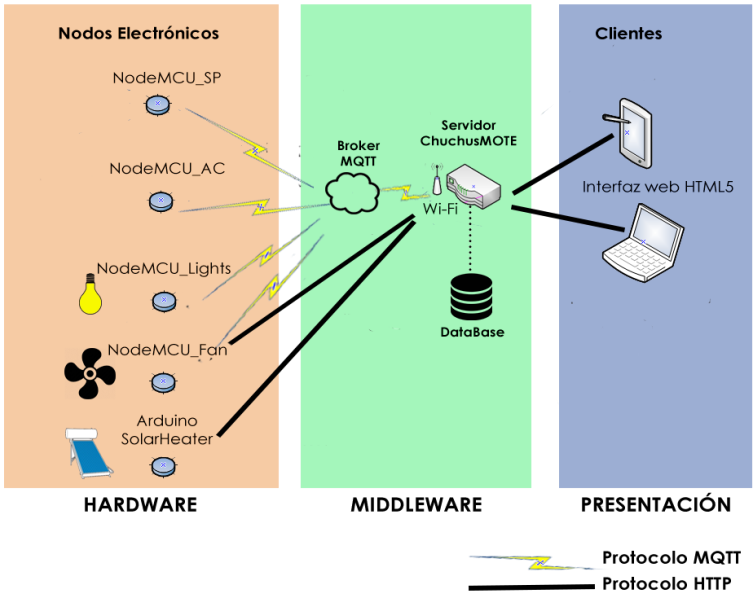

Figura 1: Arquitectura de ChuchusMOTE.

[1] ChuchusMOTE viene de la combinación de "chuchusmuti", nombre nativo del producto de la leguminosa andina llamada "tarwi" (*Lupinus mutabilis)*, y "MOTE", nombre en inglés con el que se refiere comúnmente a los sensores IoT.

2.1. Nivel Hardware

Está conformado por múltiples microcontroladores (Micro Controller Unit o CMU), capaces de adquirir señales a través de sus periféricos. El sistema cuenta con el microcontrolador ESP8266 [4], que es un MCU de bajo costo, que incluye con toda la pila de protocolos TCP/IP y una capacidad para transmitir datos vía Wi-Fi. Se implementaron cuatro módulos diferentes sobre placa de desarrollo NodeMCU[2], cada uno con una función específica:

a) *NodeMCU_SP* : Recolección variables de paneles solares y de un piranómetro
b) *NodeMCU_AC* : Medición de corriente AC
c) *NodeMCU_Lights* : Control para encendido y apagado de luces
d) *NodeMCU_Fan* : Control para encendido/apagado de un extractor de aire

Asimismo el sistema cuenta con una placa Arduino (*Arduino_SolarHeater*), que se ocupa de enviar la temperatura tomada de varios sensores y decide el encendido/apagado de un calefón solar.

2.2. Nivel Middleware

La capa middleware de ChuchusMOTE cuenta con los siguientes componentes:

a) Una aplicación web que implementa la lógica del sistema
b) Una base de datos para almacenar los datos generados por los dispositivos del nivel hardware.
c) Un servicio de comunicación ligero para envío de datos de manera asíncrona.

Antes de describir estos componentes, observemos que la comunicación entre los diferentes elementos del sistema sigue un modelo cliente-servidor, utilizando los protocolos HTTP (Hypertext Transport Protocol) [5] y MQTT (Message Queueing Transport Protocol) [6], de la siguiente manera:

- **Uso del protocolo HTTP**
  El protocolo HTTP, que es el protocolo de base de la tecnología Web, incluye una serie de métodos de petición para la interacción entre cliente y servidor. ChuchusMOTE utiliza una interface de tipo RESTful [7] simplificada que utiliza dos métodos: GET para solicitar información al servidor y POST para enviar información al servidor. El nodo electrónico Arduino_SolarHeater utiliza el interface REST para enviar información leída de sus sensores a través de la API (Application Programming Interface) desarrollada en el servidor ChuchusMOTE, y a su vez implementa un mini-servidor REST para recibir datos del servidor. El nodo NodeMCU_Fan también utiliza el interface REST para solicitar información sobre el modo de operación y temperatura límite (o "set point").

- **Uso del protocolo MQTT**
  El protocolo MQTT que está basado en el patrón "publicación/suscripción" o "publish/subscribe", es simple y ligero, diseñado para dispositivos con restricciones y con poco ancho de banda disponible para su comunicación, alta latencia o no confiables [8]. Este protocolo es por ende idóneo para aplicaciones IoT. Como muestra la Figura 2, las interacciones entre elementos de tipo "Publisher" y "Subscriber" se realiza a través de una entidad intermedia (llamada *Broker MQTT*), que recibe eventos publicados organizados por "tópicos", y los distribuye a aquellos suscritos a dichos tópicos. La comunicación es totalmente asíncrona, por lo que no se require tener explicitamente un canal de comunicación abierto constantemente. Adicionalmente MQTT soporta la noción de QoS (Quality of Service) que permite enviar mensajes siguiendo tres niveles de comunicación: 0 (at most once), 1 (at least once) y 2 (exactly once), indicando respectivamente que un mensaje será enviado sin garantias que llegue (best effort), llegará al menos una vez (pero puede ser duplicado), y llegará exactamente una vez sin duplicados.

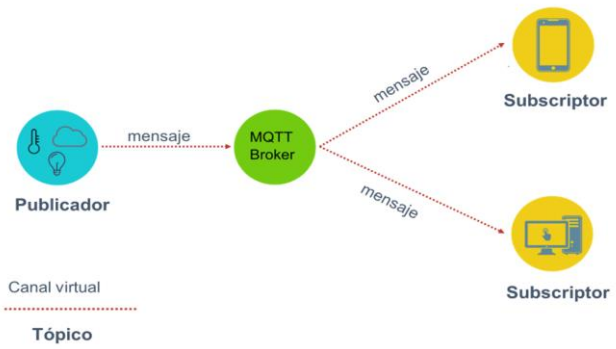

Figura 2: Comunicación "publish/subscribe" asíncrona en MQTT.

[2] http://nodemcu.com/

ChuchusMOTE utiliza MQTT para enviar datos desde los diferentes nodos electrónicos. El servidor se suscribe a diferentes tópicos hacia los cuales los diferentes nodos envían los datos. Para el envío de datos no críticos (e.g. datos de sensores de temperatura), se utiliza un nivel de QoS de 0, puesto que la pérdida de datos no representa un problema. Sin embargo, para el envío de información de crítica de control, se utiliza un nivel de QoS de 2.

2.2.1. Servidor ChuchusMOTE

El aplicación Web del Servidor ChuchusMOTE utiliza HTTP para comunicarse con los clientes Web (en la capa Presentación), y con el nodo Arduino a través del interface REST. La aplicación Web fue implementada utilizando la pila MEAN [3] (MongoDB, Express, AngularJS y Node.js), ampliamente basada en el lenguaje de programación JavaScript, ideal para el manejo de eventos asíncronos. El servidor, además de responder a las solicitudes HTTP de los clientes Web, también incluye un módulo MQTT para suscribirse al Broker MQTT y recibir las notificaciones asíncronas. Finalmente, para enviar los datos en tiempo-real a los clientes Web, el servidor ChuchusMOTE utiliza el módulo WebSocket socker.io[4] que permite abrir un canal bidireccional entre el servidor y el cliente, sin necesidad de cargar el contenido de toda la página de la aplicación en el cliente Web.

2.2.2. Base de Datos

ChuchusMOTE utiliza una base de datos MongoDB que es una base de datos NoSQL (i.e. no estructurada) y orientada a documentos [9]. El servidor ChuchusMOTE se conecta al servidor MongoDB utilizando el modulo "mongoose" de Node.js. MongoDB almacena datos en documentos similares a JSON[5] de manera flexible, lo que significa que los campos pueden variar de documento a documento y la estructura de datos se puede cambiar con el tiempo. Cada variable almacenada en la base de datos tiene un modelo propio lo que facilita la búsqueda y organización de la información solicitada por los clientes web.

Los datos enviados por los nodos electrónicos, son almacenados en el servidor MongoBD en forma de documentos JSON con la siguiente estructura:

```
{
   "_id" : ... ,
   "sensorID" : ... ,
   "sensorVAL" : ... ,
   "timestamp" : ...
}
```

Con la siguiente correspondencia:

- **`_id` :** Identificador único (llave primaria) de cada documento en la base de datos.
- **`sensorID` :** Correspondiente al identificador del sensor del nodo electrónico que envía los datos, y que se usa como ID el tópico MQTT.
- **`sensorVAL` :** Destinado a guardar los valores que envían los sensores
- **`timestamp` :** Contiene la fecha en la cual se guardaron los datos en la base de datos. El valor corresponde al standard *Unix epoc time*, i.e. milisegundos desde 01/01/1970 hasta el instante en que los datos son almacenados dentro de la base de datos.

2.2.3. Broker MQTT

Para la distribución de los mensajes MQTT se utilizó el Broker Open Source Eclipse MOSQUITTO[6] que implementa el protocolo MQTT 3.1.1, que es el estándar utilizado en ChuchusMOTE.

Si bien cada componente del nivel middleware pueden desplegarse en servidores físicos diferentes, la aplicación Web del Servidor ChuchusMOTE, la base de datos MongoDB, y el Broker MQTT fueron instalados en un mini-computador RaspberryPi 3 Modelo B[7], de manera que el sistema sea más compacto y de bajo costo. Los tres componentes, pueden funcionar de manera separada con un mínimo de esfuerzo de re-configuración.

2.3. Nivel Presentación

En el nivel de presentación de ChuchusMOTE, se implementó una aplicación interactiva utilizando el lenguaje de programación JavaScript y HTML con hojas de estilo CSS para desplegar una interfaz amigable al usuario. La

3 http://mean.io/
4 https://socket.io/
5 JavaScript Object Notation: es un formato de texto ligero para el intercambio de datos
6 https://mosquitto.org/
7 https://www.raspberrypi.org/

interacción entre el servidor y cliente se realiza a través varios scripts que son encargados de permitir desplegar distintas funciones dinámicas. Para el despliegue de gráficos se utilizó la librería AmCharts[8] que permite la visualización interactiva de datos en diferentes tipos de vistas (e.g. line charts, pie chart). Para la recepción de datos en tiempo real, el interface Web cliente utiliza una conexión bidireccional WebSocket socket.io, en codigo embedido en las páginas Web.

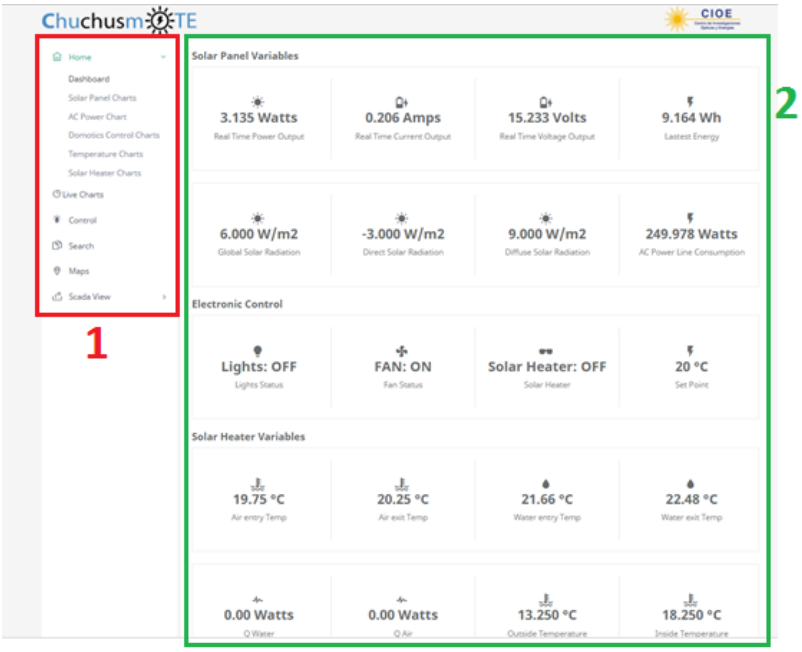

Figura 3: Interfaz gráfica de ChuchusMOTE.

La Figura 3 muestran los 2 sectores de la interfaz gráfica Web: el sector (1) contiene todo el menú de navegación y opciones del sistema (panel de control, visualización de datos históricos, visualización por tipo, visualización de datos en tiempo-real y búsqueda), y el sector (2) contiene el marco de visualización de datos.

3. DISEÑO Y DESARROLLO DEL SISTEMA

A continuación se describe la estructura del servidor ChuchusMOTE y la interacción entre los diferentes módulos. Asimismo, describimos en detalle la comunicación entre los componentes y como se combinaron los protocolos de comunicación para el envío de mensajes de datos y control. Finalmente, describimos la implementación de los dispositivos electrónicos desarrollados, su lógica de funcionamiento y cómo interactúan con el resto del sistema.

3.1. ESTRUCTURA DEL SERVIDOR CHUCHUSMOTE

La estructura y diferentes módulos que conforman el servidor ChuchusMOTE se muestra en la Figura 4.

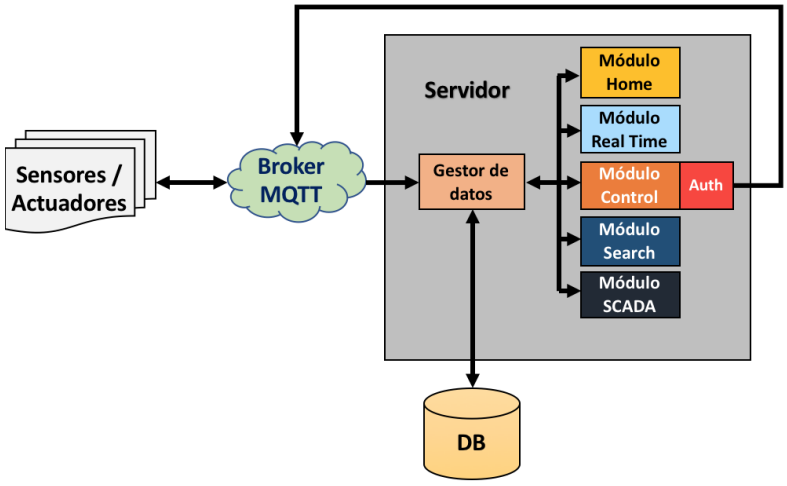

Figura 4: Estructura del servidor ChuchusMOTE.

[8] https://www.amcharts.com/

El servidor ChuchusMOTE integra 5 módulos usuario: *Home* (página principal de la aplicación), *Real Time* (visualización de datos en tiempo real), *Control* (interacción con los nodos), *Search* (búsqueda de datos históricos) y *SCADA*[9] (representación gráfica del sistema). El componente *Gestor de datos*, interactúa con los 5 módulos de manera directa haciendo de interface con la base de datos y el Broker MQTT (e.g. para búsqueda de datos históricos o visualización en tiempo real respectivamente). Cuando el módulo *Control* es desplegado, éste se conecta al Broker MQTT para poder enviar eventos de control a los diferentes nodos electrónicos que previamente se han suscrito a los tópicos de control correspondientes. El módulo *Control* requiere autentificación de usuario previa, para poder realizar operaciones remotas de control domótico. El componente *Auth* gestiona la autentificación de usuarios.

3.2. FUNCIONAMIENTO Y COMUNICACIÓN

La Figura 5 muestra el diagrama de funcionamiento del sistema completo, donde para simplificar se describe la comunicación entre un nodo sensor genérico NodeMCU, el servidor Web y el cliente Web.

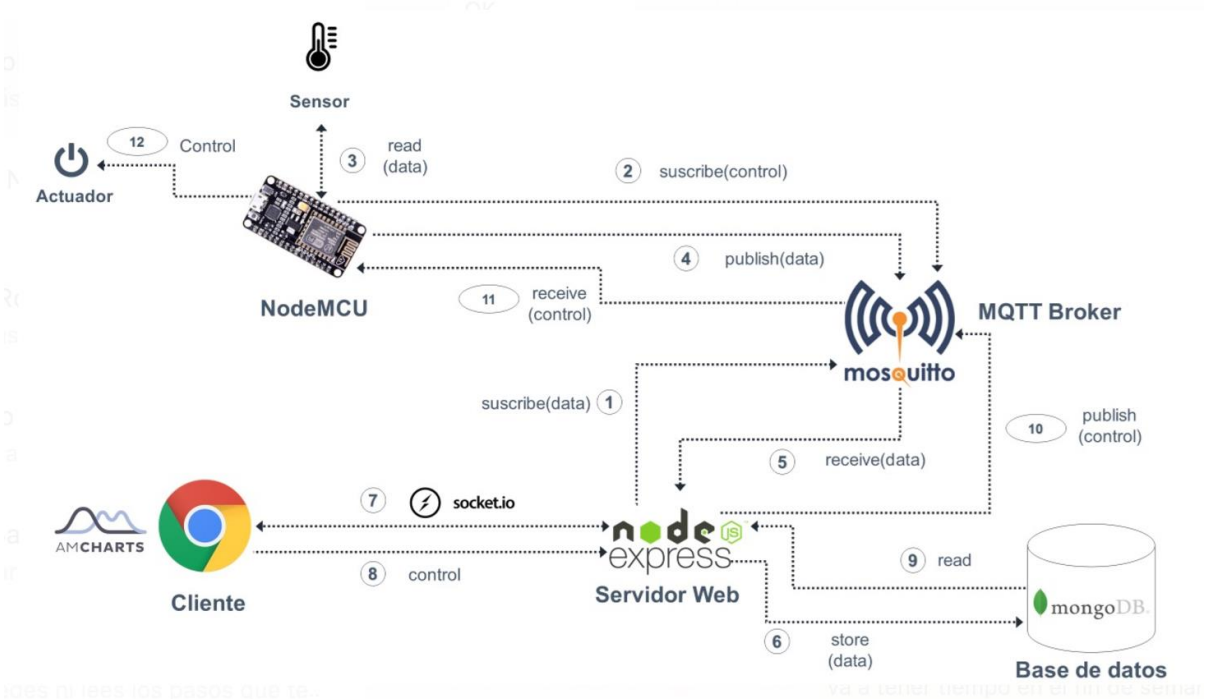

Figura 5: Diagrama de funcionamiento del sistema ChuchusMOTE.

El servidor ChuchusMOTE inicia suscribiéndose a todos los "tópicos de datos" a través de los cuales recibirá información de los diferentes NodeMCU (1). Aquellos NodeMCU que requieran recibir mensajes de control para actuadores, se suscriben a su vez a los "tópicos de control" correspondientes (2). Para el envío de datos, el NodeMCU obtiene el valor leído del sensor al que se encuentra conectado (3). Luego, el nodo genera un mensaje en formato JSON con los datos y los publica en el tópico MQTT respectivo (4) y el servidor recibe los mensajes según el tópico al que se subscribió (5). El servidor procesa el mensaje y lo almacena en la base de datos MongoDB (6) sin efectuar ningún cambio de formato adicional, resultando en una operación eficiente de escritura. Cuando un cliente Web se conecta al Servidor ChuchusMOTE, éste responde con la aplicación gráfica interactiva Web y abre un canal WebSocket socket.io (7) para el envío y recepción de datos de manera bidireccional. Si el usuario elige la visualización en tiempo-real, todo mensaje MQTT que llegue posteriormente al servidor es enviado al cliente Web por el WebSocket, mostrando la actualización de datos. Si el usuario accede al módulo de control (previa autentificación), podrá enviar comandos de control al NodeMCU correspondiente a través de servidor (8). Para esto, el servidor genera un nuevo mensaje de control en formato JSON, si es necesario lee información de la base de datos (9) y finalmente publica el mensaje MQTT en el tópico correspondiente (10). Los mensajes de control serán recibidos por el NodeMCU con subscripción al tópico (11), el cual lo procesará y realizará la operación de control en el actuador correspondiente (12).

Si bien la publicación de mensajes desde los NodeMCU no requiere algún tipo de seguridad en particular, aquellos de control si requieren ser protegidos. Además de la protección de autentificación en el módulo *Control* de la aplicación Web (ver Figura 4), se protegió la comunicación de control en las pruebas de laboratorio con autentificación básica (usuario/contraseña), y para el despliegue en la red abierta utilizando MQTT sobre el protocolo criptográfico TLS/SSL [10].

[9] SCADA significa en inglés *Supervisory Control and Data Acquisition* (Supervisión, Control y Adquisición de Datos)

3.3. DESARROLLO DE LOS NODOS ELECTRÓNICOS

Se desarrollaron cinco nodos electrónicos para el sistema ChuchusMOTE: dos exclusivamente para la recolección de datos con sensores y tres actuadores con sensores. Las TABLA 1 y TABLA 2 describen todos los nodos electrónicos con su función, sensores, actuadores y las variables medidas.

TABLA 1 - NODOS SENSORES DEL SISTEMA

Nodo	Descripción del nodo	Sensor	Variable Medida	Descripción sensor
NodeMCU_SP	Recolecta todas las variables de los paneles solares y del piranómetro	ACS712	Corriente	Mide la corriente eléctrica máxima del panel solar
		Divisor Voltaje	Voltaje	Mide la tensión eléctrica del panel solar
		Piranómetro	Radiación Solar	Mide la radiación solar
			Potencia Eléctrica	Producto de la corriente por el voltaje
			Energía Eléctrica	Integral de la potencia generada en una hora
NodeMCU_AC	Mide la corriente AC	ACS712	Corriente AC	Mide la corriente AC eléctrica

TABLA 2 - NODOS ACTUADORES DEL SISTEMA

Nodo	Descripción del nodo	Actuador	Sensor	Variable Medida	Descripción Sensor	Descripción Actuador
NodeMCU_Lights	Recibe órdenes del servidor de encendido y apagado de luces	Relay 1 canal		Ninguna		Abre o cierra el canal de energización de las luces
NodeMCU_Fan	Recibe un set point de temperatura y decide el encendido o apagado del extractor de aire	Relay 1 canal	DS18B20	Temperatura	Mide la temperatura exterior	Abre o cierra el canal de energización del extractor de aire
Arduino_SolarHeater	Envíía 5 temperaturas tomadas por distintos sensores, y decide el encendido o apagado del calefactor solar		DS18B20	Temperatura	Mide la temperatura ambiente	
			DS18B20	Temperatura	Mide la temperatura de entrada de agua a la bomba	
			DS18B20	Temperatura	Mide temperatura de salida de agua del sistema de calefacción (radiador)	
			DS18B20	Temperatura	Mide la temperatura de entrada de aire al ventilador	
			DS18B20	Temperatura	Temperatura de salida de aire de la superficie del radiador	
		Relay 1 canal				Abre o cierra el canal de energización del calefactor solar

A continuación se describe cada uno de los nodos, y el funcionamiento del software embebido desarrollado para cada uno. La programación del nodo se realizó en el lenguaje C++ dentro del Arduino IDE[10] que permite añadir modulos de software a la placa de desarrollo NodeMCU con el microcontrolador ESP826.

3.3.1. Nodo electrónico NodeMCU_SP

El nodo NodeMCU_SP está encargado de recolectar información proveniente de varios sensores relativos a paneles solares. La Figura 6 muestra en la parte superior el esquema de conexión del NodeMCU_SP y la parte inferior el desarrollo electrónico correspondiente. Las variables recolectadas de los paneles solares son las siguientes:

- Voltaje: Variable medida por un divisor de voltaje
- Corriente: Variable medida por el sensor ACS712
- Potencia: Variable resultante de la multiplicación de Voltaje por Corriente
- Energía: Integral de la potencia entregada por los paneles durante una hora

[10] https://www.arduino.cc/

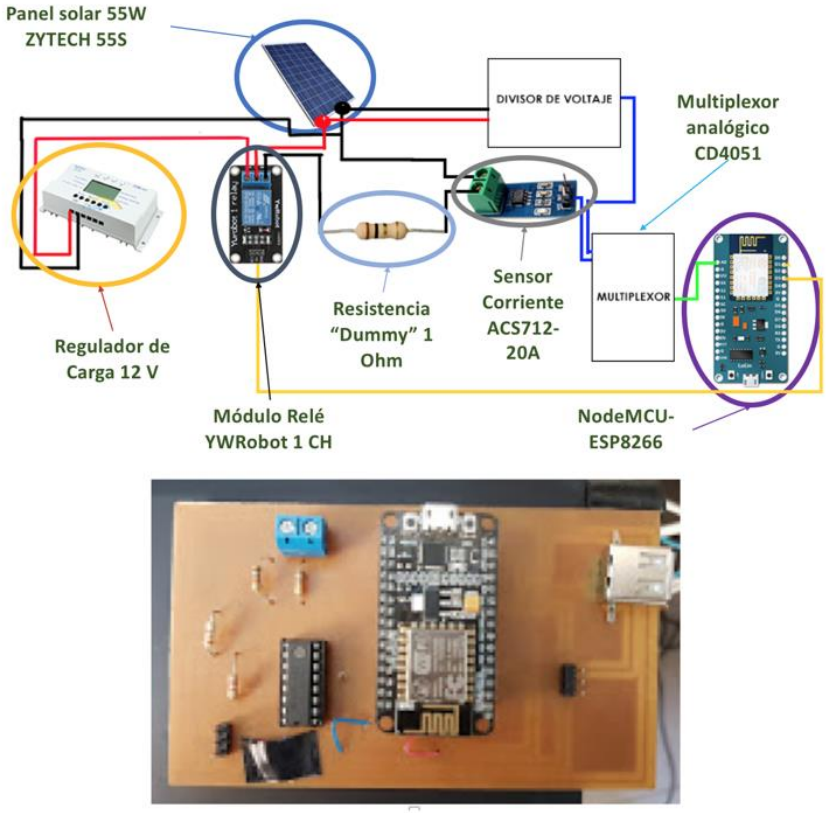

Figura 6: Esquema de conexión del NodeMCU_SP (parte superior) y el desarrollo electrónico (parte inferior).

TABLA 3: TÓPICOS MQTT PARA LOS DIFERENTES NODOS

Nodo	Tópico MQTT	Tipo	Descripción
NodeMCU_SP	`panels/amp`	Datos	Corriente DC proveniente de los paneles solares
	`panels/voltage`	Datos	Voltaje DC proveniente de los paneles solares
	`panels/power`	Datos	Potencia DC proveniente de los paneles solares
	`panels/energy`	Datos	Energía por hora proveniente de los paneles solares
NodeMCU_AC	`ac/power`	Datos	Potencia eléctrica AC eficaz
NodeMCU_Lights	`control/lights`	Control	Comando ON/OFF a las luces AC
NodeMCU_Fan	`temp/exterior`	Datos	Temperatura exterior al Laboratorio
	`temp/interior`	Datos	Temperatura ambiente del Laboratorio
	`control/fan`	Control	Comando ON/OFF al extractor de aire

Al iniciarse, el nodo NodeMCU_SP recibe los parámetros de conexión hacia el enrutador Wi-Fi, y la dirección del Broker MQTT. Luego, procede a la lectura de los datos de los sensores y debido a que se usa un multiplexor analógico, primero se envía un valor digital al multiplexor para habilitar la salida analógica a cada sensor. Posteriormente realiza el envío de los datos leído al Broker MQTT sobre los tópicos respectivos a cada sensor, i.e. `panel/amp`, `panel/voltage`, `panels/power` y `panels/energy` del nodo NodeMCU_SP como muestra la TABLA 3.

3.3.2. Nodo electrónico NodeMCU_AC

El nodo NodeMCU_AC está encargado de realizar la medición del consumo eléctrico. La Figura 7 muestra el desarrollo electrónico del nodo, que incluye el sensor de corriente eléctrica ACS712 mostrado en el recuadro. Una vez los parámetros MQTT inicializados y la conexión Wi-Fi establecida, el nodo realiza la lectura del valor de intensidad de corriente eléctrica utilizando el sensor ACS712, y luego se calcula la potencia eléctrica eficaz como el *valor cuadrático medio* o RMS (del inglés *root mean square*). El algoritmo usado en este nodo consiste en realizar mediciones durante medio segundo, lo que equivale a realizar mediciones durante 25 ciclos (señal de 50Hz). Durante este tiempo se obtienen las lecturas máximas (*Imax*) y mininas (*Imin*), y se calcula la intensidad de corriente pico (*Ip*) como el promedio de ambas lecturas. A este resultado le restamos la amplitud del ruido que está presente cuando la corriente es 0. Con la corriente pico se procede a calcular la intensidad de corriente en RMS y la potencia eléctrica en RMS. Estos valores son finalmente publicados en un mensaje MQTT en el tópico `ac/power` del nodo NodeMCU_AC como muestra la TABLA 3. El nodo entra modo reposo durante 1 minuto, y vuelve a leer los datos y publicarlos.

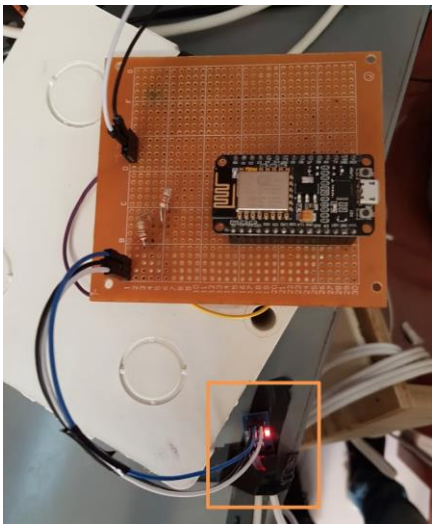

Figura 7: Desarrollo electrónico del NodeMCU_AC.

3.3.3. Nodo electrónico NodeMCU_Lights

El nodo NodeMCU_Lights está encargado del encendido y apagado de luces, a través de la recepción de mensajes MQTT. La Figura 8 muestra el desarrollo electrónico del nodo NodeMCU_Lights. Cuando llega el mensaje de control sobre el protocolo MQTT en el tópico `control/lights` (ver TABLA 3), y dependiendo del mensaje recibido, el nodo envía una acción sobre un actuador que controlada las luces (para cambiar su estado).

Figura 8: Desarrollo electrónico del NodeMCU_Lights.

3.3.4. Nodo electrónico NodeMCU_Fan

El nodo NodeMCU_Fan está encargado del control de un extractor de aire. La Figura 9 muestra el desarrollo electrónico del NodeMCU_Fan.

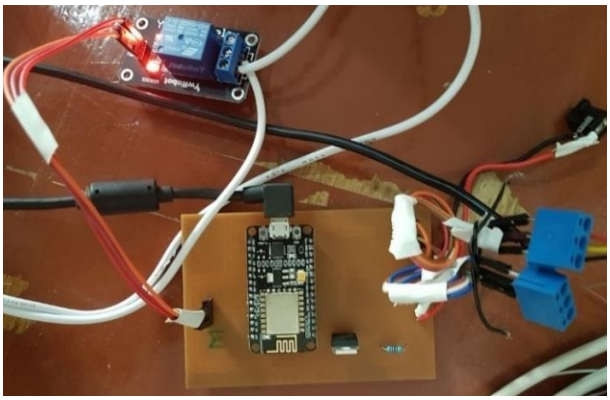

Figura 9: Desarrollo electrónico del NodeMCU_Fan.

Este nodo funciona en dos modos: manual o automático (según el modo que el usuario escogió en el Modo de Control). Al inicio, el NodeMCU_Fan utiliza el interface REST del servidor ChuchusMOTE para solicitar el último dato de modo de ejecución guardado en la base de datos. En modo manual, el nodo NodeMCU_Fan se suscribe al tópico `control/fan` (ver TABLA 4 - INTERFACE REST DE CHUCHUSMOTETABLA 4) y espera las ordenes de encendido/apagado posteriores.

En el modo automático el "set point" (i.e. el valor límite en °C) es devuelto en la respuesta de la solicitud REST. Luego, el nodo NodeMCU_Fan lee los valores de temperatura ambiente (interior) y exterior de los sensores, y dependiendo de la diferencia con el "set point", i.e. si la diferencia entre exterior e interior sobrepasa el "set point", procede accionar el actuador enviando una señal positiva de voltaje y cerrando el canal de comunicación eléctrico entre la línea de alimentación de tensión alterna y el extractor de aire. Los valores de temperatura exterior e interior son publicados en el tópico MQTT `temp/exterior` y `temp/interior`, respectivamente.

3.3.5. Nodo electrónico Arduino_SolarHeater

El nodo Arduino_SolarHeater (ver Figura 10) está encargado del encendido y apagado automático del calefactor solar, y la lectura de 4 valores de temperatura:

- `Tin_water`: Temperatura en °C de entrada de agua al radiador
- `Tout_water`: Temperatura en °C de salida de agua del sistema de calefacción solar
- `Tin_air`: Temperatura en °C de entrada de aire al ventilador
- `Tout_air`: Temperatura en °C de salida de aire de la superficie del radiador

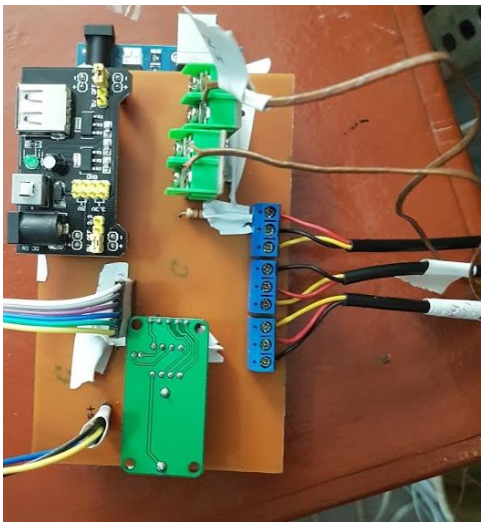

Figura 10: Desarrollo electrónico del Arduino_SolarHeater.

A diferencia de los otros nodos, el Arduino_SolarHeater utiliza una conexión Ethernet por cable. El nodo solicita al servidor el "set point" guardado en la base de datos a través de su interface REST, y compara el valor recibido al de los sensores de temperatura interior. En caso de ser inferior, se enciende el calefactor solar enviando un 0 lógico al actuador. Todos los datos leídos de temperatura y de calor calculados al servidor a través del interface REST correspondiente. La TABLA 4 muestra el interface REST de base de ChuchusMOTE.

TABLA 4 - INTERFACE REST DE CHUCHUSMOTE

Interface REST	Método	Descripción
`sensor/{VAR_OUT}/id/:id` **`sensor/{VAR_OUT}`**	GET	Búsqueda de datos guardados en la DB (con o sin `id`), para valores de `{VAR_OUT}`: `V` (voltaje), `C` (corriente), `P` (potencia), `E` (energía), `R` (radiación solar), `TInt` (temperatura interna) y `TExt` (temperatura externa), `TWInt` (temperatura de agua interior), `TWExt` (temperatura de agua exterior), `TFInt` (temperatura salida de aire), `TFExt` (temperatura entrada de aire), `Fan` (estado extractor de aire), `Lights` (estado luces), `Solarh` (estado calefactor solar), `ACP` (potencia AC), `SP` (valor "set point")
`sensor/{VAR_OUT}/from/:from/to/:to`	GET	Búsqueda temporal para valores de `{VAR_OUT}` entre fechas `:from` y `:to`
`sensor/last/{VAR_OUT}`	GET	Búsqueda del ultimo valor de la variable `{VAR_OUT}`
`sensor/{VAR_IN}/id/:id`	POST	Inserta el valor en la DB para los siguientes valores de `{VAR_IN}`: `TWInt` (temperatura de agua interior), `TWExt` (temperatura de agua exterior), `TFInt` (temperatura salida de aire), `TFExt` (temperatura entrada de aire), `ShStatus` (estado ON/OFF del calefactor solar), `QAir` (calor del aire) y `QWater` (calor del agua)

4. RESULTADOS

Se instaló el sistema completo e hicieron pruebas de validación de los diferentes componentes del sistema. El sistema se encuentra totalmente funcional y los datos recolectados están disponibles en el portal de ChuchusMOTE[11] donde se puede interactuar con el sistema domótico, analizar los datos recolectados durante varios meses, y visualizar los datos en tiempo real los diferentes datos generados por distintos sensores, como muestra la Figura 11.

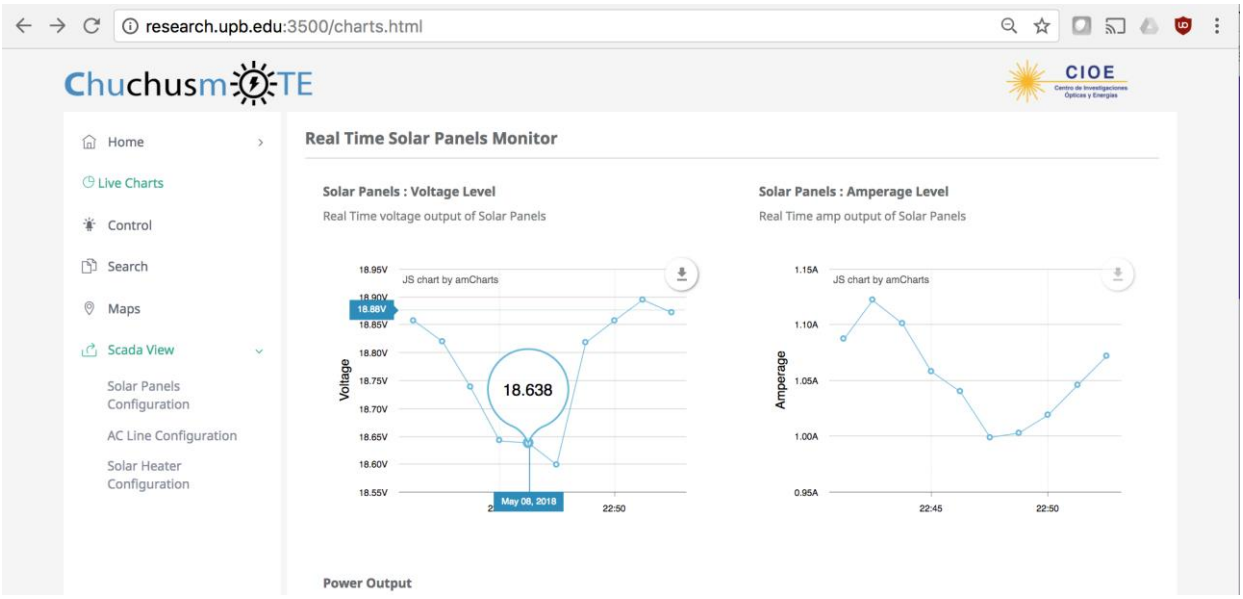

Figura 11: Captura de pantalla del portal del sistema ChuchusMOTE.

Para la validación del sistema, que duró 4 semanas, se verificó que los nodos electrónicos comunican correctamente los datos al servidor, son guardados en la base de datos, y que los valores corresponden calculados corresponde a las especificaciones de los dispositivos en observación y controlados. Como ejemplo, a continuación mostramos la validación correspondiente a medidas tomadas el mismo día (27/07/2017).

En la parte solar, las mediciones fueron realizadas de un panel solar marca ZYTECH-ZT55S que tiene una potencia máxima teórica de 55W y una eficiencia máxima de 13%. La Figura 12 muestra la gráfica interactiva de medición del voltaje del panel solar observamos que el voltaje máximo medido es de aproximadamente 19.7 Voltios, y se observa claramente la curva característica correspondiente a la radiación solar de un día de invierno entre las 6:30 y las 18:00 aproximadamente.

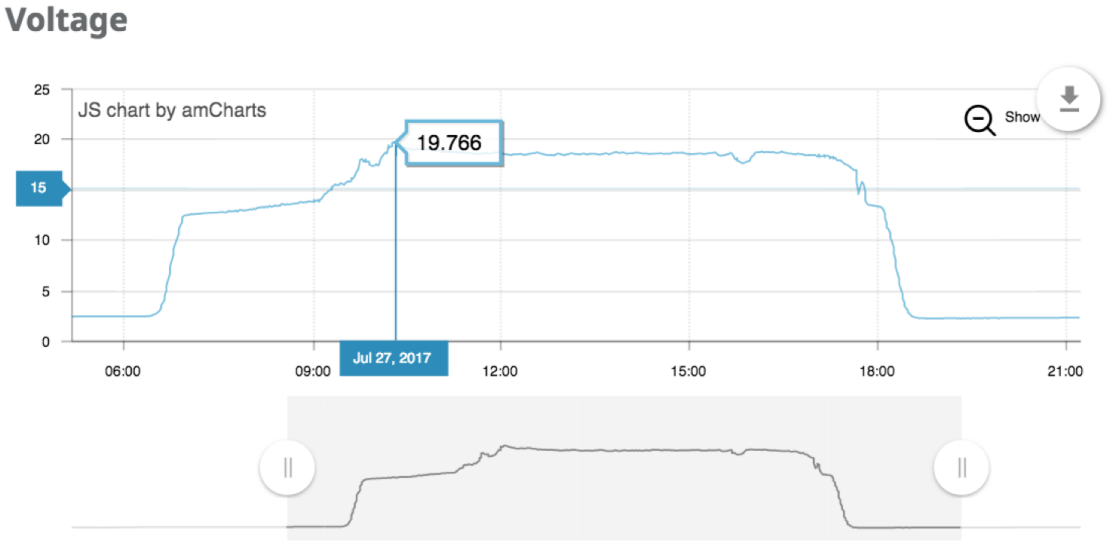

Figura 12: Voltaje del panel fotovoltaico.

[11] http://research.upb.edu:3500

La Figura 13 muestra la gráfica de medición de la intensidad de corriente del panel solar, observamos que la corriente máxima medida es de aproximadamente 2.6 A. Para la validación la medidas de potencia del panel solar, se evaluaron todos los datos recibidos durante un día soleado, como muestra la Figura 14 entre las 09:00 y las 18:00, donde se observa un incremento de potencia de salida del panel solar alcanzando un máximo de 50.888 Watts. Considerando que el panel solar utilizado por el sistema un ZYTECH-ZT55S que tiene una potencia máxima teórica de 55W según su hoja de datos[12], se observa que no alcanza el máximo de potencia señalado. Sin embargo, la medición realizada por el nodo electrónico está muy cerca de dicho máximo teórico. Para entender esta diferencia, es necesario considerar que las mediciones realizadas por los sensores del nodo tienen un margen de error relativo del 1% para lecturas de voltaje y 5% para lecturas de intensidad de corriente. Además, se tiene que tomar en cuenta que existe un cierto deterioro inherente en las celdas fotovoltaicas debido a la antigüedad de los paneles utilizados (de aproximadamente 5 años).

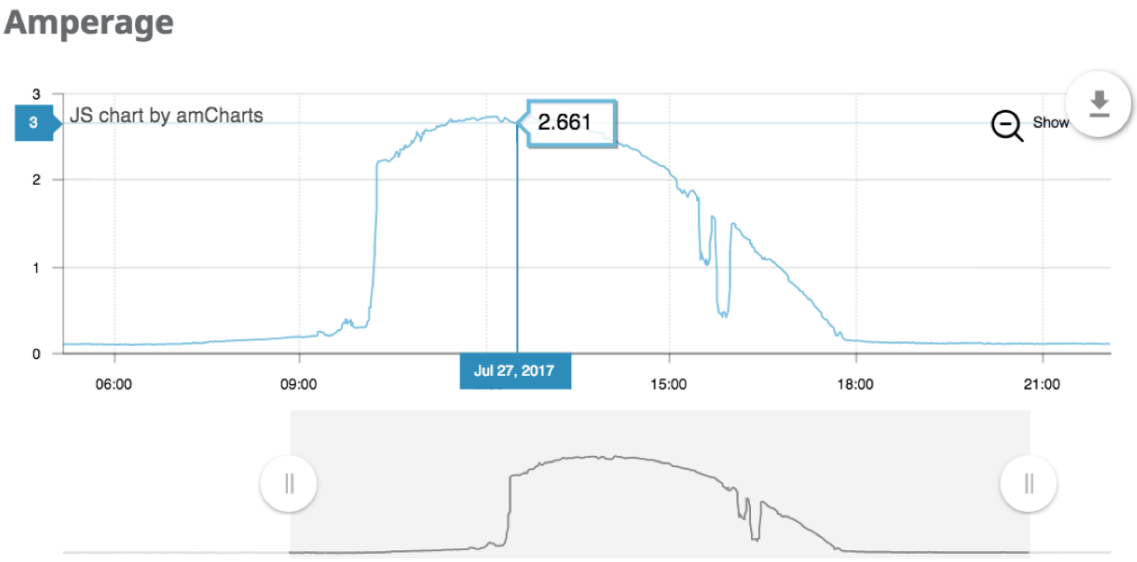

Figura 13: Corriente del Panel Solar Fotovoltaico.

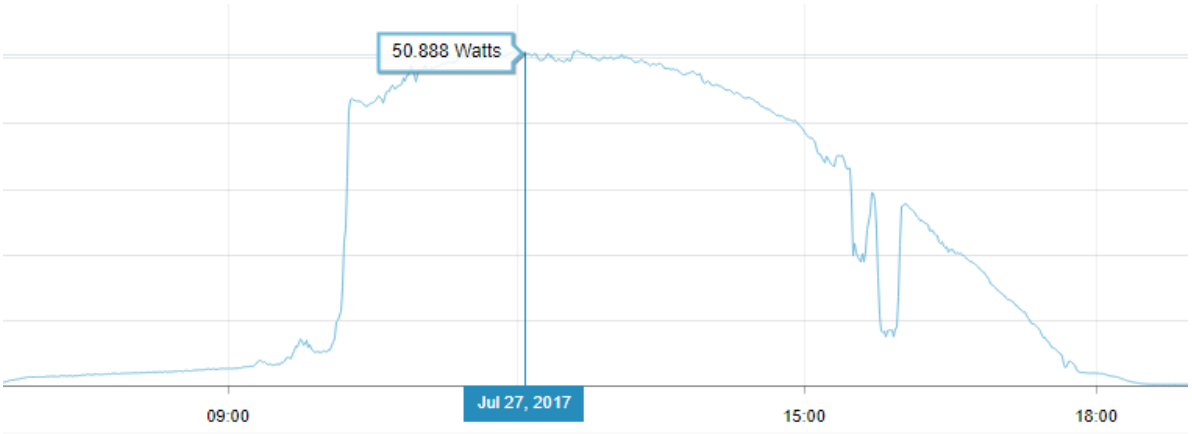

Figura 14: Potencia del Panel Solar Fotovoltaico.

Para el cálculo de la eficiencia del sistema fotovoltaico se tomó en cuenta el área de 0.492 m^2 del panel solar y la radiación solar (W/m^2) del momento en que se realiza la medición de potencia eléctrica del panel solar. En la Figura 15 se observa la eficiencia del panel solar, donde puede ver un pico máximo de eficiencia de 12.2%. Según la hoja de datos el panel solar cuenta con una eficiencia máxima de 13%; lo que demuestra que está dentro del rango de eficiencia aceptable.

[12] http://www.posharp.com/zt55s-solar-panel-from-zytech-solar_p1925167756d.aspx

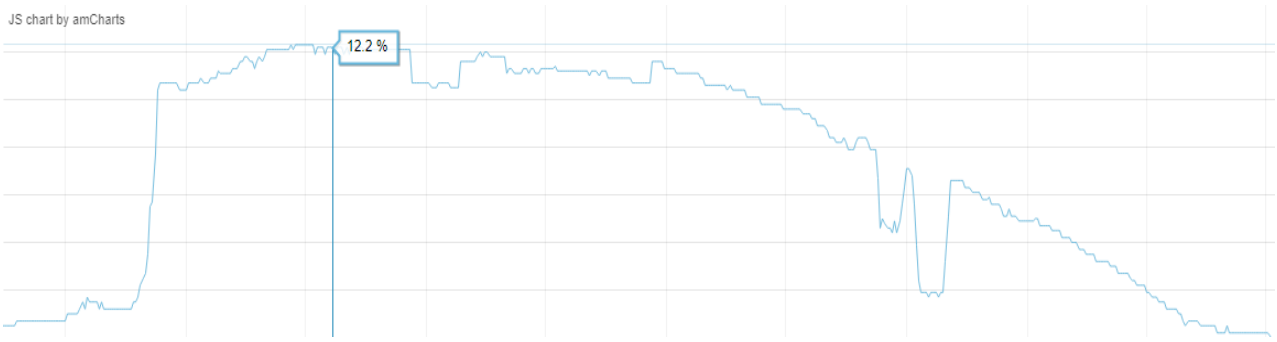

Figura 15: Eficiencia del Panel Solar Fotovoltaico.

El sistema de control domótico se observa en la Figura 16, se puede identificar todos los componentes del sistema de control: luces, extractor de aire y calefactor solar (controlado a través del "set point" definido por el usuario). Además, muestra el estado actual de todos los elementos controlados (encendido/apagado).

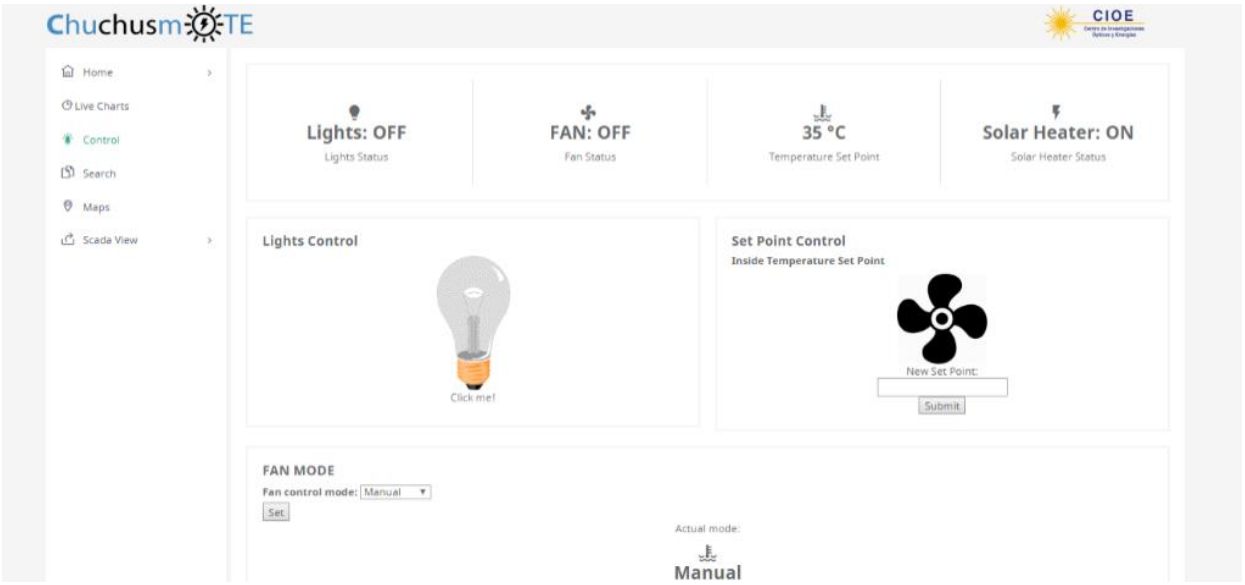

Figura 16: Sistema de control domótico de ChuchusMOTE.

5. CONCLUSIONES

En este artículo se presentó el diseño y desarrollo ChuchusMOTE, un sistema de monitoreo energético con control domótico basado en tecnología IoT, que incluye el desarrollo electrónico de una red de sensores y actuadores que comunican utilizando el protocolo de comunicación MQTT. Dicho protocolo, al estar basado en el modelo "publicación/subscripción", permite una comunicación asíncrona entre los diferentes elementos del sistema tanto para el envío de datos como mensajes de control. ChuchusMOTE cuenta con un servidor y aplicación Web con una interface de tipo REST, y permite visualizar datos en tiempo-real, realizar búsquedas por fecha, y realizar operaciones remotas de control en los actuadores, entre otros. Los resultados obtenidos confirman la factibilidad de la combinación de la tecnología IoT, la programación embebida y el desarrollo electrónico utilizando elementos electrónicos comerciales simples y de facil acceso.

6. BIBLIOGRAFÍA

[1] International Telecommunication Union, «Internet of Things Global Standards Initiative,» ITU, Julio 2015. [En línea]. Disponible: http://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx. [Último acceso: 11 Junio 2017].

[2] D. Serpanos y M. Wolf, Internet-of-Things (IoT) Architectures, Algorithms, Methodologies, Atlanta: Springer, 2018.

[3] J. Gubbi, R. Buyya, S. Marusic y M. Palaniswami, «Internet of Things (IoT): A vision, architectural elements, and future directions,» *Future Generation Computer Science*, vol. 29, nº 7, pp. 1645-1660, 2013.

[4] Espressif Systems, «ESP8266 SDK API Guide,» Espressif Systems IOT Team, 2015.

[5] Internet Engineering Task Force, «Hypertext Transfer Protocol -- HTTP/1.1,» IETF, 1999. [En línea]. Disponible:

https://tools.ietf.org/html/rfc2616. [Último acceso: 31 Mayo 2017].

[6] OASIS Consortium, «MQTT Version 3.1.1 - OASIS Standard,» 29 Octorber 2014. [En línea]. Disponible: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html.

[7] R. Fielding, Architectural Styles and the Design of Network-based Software Architectures, Irvine, USA: University of California, Irvine, 2000.

[8] Message Queue Telemetry Transport, «MQTT F.A.Q,» MQTT, [En línea]. Disponible: http://mqtt.org/faq. [Último acceso: 30 Mayo 2017].

[9] MongoDB, «What is MongoDB?,» MongoDB, [En línea]. Disponible: https://www.mongodb.com/what-is-mongodb. [Último acceso: 26 Mayo 2017].

[10] J. Viega, M. Messier y P. Chandra, Network Security with OpenSSL, O'Reilly, 2002.