ISSN 1870-4069

Ernest: sistema embebido para control de casas inteligentes mediante correos electrónicos con cifrado SSL

José Francisco Rodríguez Arellano, Emmanuel Dávila Delgado,
Mario Alberto Ruíz Durán, Martín Isaac Falcón Segovia,
Salvador Abraham Medina Rangel, Cristian Jael Mejía Aguirre

Centro de Investigación e Innovación en Tecnologías de la Información y Comunicación
Aguascalientes, Aguascalientes, México
{francisco.rodriguez,emmanuel.davila,mario.ruiz,cristian.mejia}
@infotec.mx, elontesla@gmail.com, salami.95@hotmail.com

Resumen. El funcionamiento de los servicios de domótica o servicios orientados a las SmartHouse dio pie a la aparición de Ernest, aplicación orientada a transformar una vivienda o construcción con acceso a internet y a electricidad en una casa inteligente. Ernest hace uso de los servicios de correo electrónico sin necesidad de tener un contrato con una empresa que permita conectarse a la nube. La implementación de un circuito electrónico lleva a cabo esta tarea; en este artículo se incursiona en la arquitectura de este circuito que integra el Internet de las cosas y habilita o deshabilita cuatro tomacorrientes. Se presentan también el desarrollo de tres algoritmos que permitan acceder y usar los correos electrónicos para establecer comunicación usuario-tarjeta y viceversa, con la finalidad de controlar o informar del estado en que se encuentran sus tomacorrientes, así como filtrar los e-mails recibidos. Estos algoritmos permiten que el sistema embebido se conecte a internet haciendo uso del protocolo de seguridad SSL y agilicen la lectura de los nuevos correos electrónicos en la bandeja de entrada y, en caso de ser necesario, el envío de los estados de los tomacorrientes.

Palabras clave: internet de las cosas, domótica, sistemas embebidos.

Ernest: Embedded System to Control Smart Homes through Emails with SSL Encryption

Abstract. The performance of home automation services or SmartHouse-oriented services gave rise to the emergence of Ernest, an application aimed at transforming a home or building with access to the Internet and electricity in a smart home. Ernest makes use of email services without hire a company that allows to connect to the cloud. The implementation of an electronic circuit accomplishes this task; In this paper, we explore the architecture of this circuit that integrates the Internet of Things and enables or disables four power outlets. There is also the development of three algorithms that allow access and use of

emails to establish user-card communication, in order to control or report the status of their power outlets, as well as filter the e-mails received. These algorithms allow the embedded system to connect to the internet using the SSL security protocol and streamline the reading of new emails in the inbox and, if necessary, the sending of the current status of the power outlets.

Keywords: internet of things, domotics, embedded systems.

1. Introducción

Hoy la mayoría de los servicios para SmartHouse venden una aplicación que permite el acceso a la nube a través de la cual se ponen en contacto con las cosas, desde sensores hasta encendido de luces. Esto implica otorgarle información personal a una empresa que debe contratarse para la instalación o en caso de asistencia técnica lo que las pone en la mira de los hackers; por tal motivo, empresas como SmartThings [1] han optado por hacer uso de las redes sociales con la finalidad de evitar el hurto de información, bien sea de los usuarios o de la empresa misma.

Sin embargo, el despliegue de comunicación automática de objetos en nuestras vidas, de manera natural, representa un peligro para el futuro al poner en riesgo nuestra información personal [2]. Esto se debe a que la información en la nube trae grandes interrogantes y algunos problemas, dado que la información es vulnerable a ataques de hackers, o frente a malware o virus por lo que se buscan los errores de comunicación y las posibles amenazas a la seguridad tanto en la protección de datos, como en la privacidad de los usuarios [3,4,5]. En consecuencia, los ataques de seguridad se vuelven sumamente problemáticos para el IoT amenazándolas desde la accesibilidad física a sensores, actuadores y objetos, hasta la apertura de los sistemas donde se incluye el hecho de que la mayoría de los dispositivos se comunican de forma inalámbrica [6]. Por estas razones, entre otras, se vuelve obligatorio hacer uso de los estándares y protocolos; los estándares permiten aumentar la calidad del producto (es decir, cumplen con los requisitos de rendimiento y seguridad) y garantizan la interoperabilidad de varios componentes en un sistema [5].

Por lo anteriormente dicho, se vuelve necesario definir el término "correo electrónico" o e-mail, aunque para muchos expertos del tema es suficiente decir que no es más que una recreación del servicio de correo tradicional sobre la base de un medio de comunicación moderno y digital [7]. Y es que la mayoría de los e-mails en Internet se transfieren por el Protocolo de Transferencia Simple de Correo (SMTP) [8] y, para recuperarlos de los usuarios finales de almacenamiento, es necesario acudir a uno de los protocolos POP3 (Protocolo de Oficina, versión 3 [9]) o IMAP4 (Protocolo de Acceso a Correo de Internet versión 4 [10]). Ambos protocolos están basados en texto con roles distintos de clientes y servidores y son servicios de mensajería tradicional sobre la base de un medio moderno y digital de comunicación, pertenecientes a la familia de MHS (Sistemas de Manejo de Mensaje). Estos protocolos transportan los mensajes a través del MTS (Sistema de Transportación de Mensaje) –subsistema del MHS– y permiten la interacción con el usuario a través del UA (Agentes de Usuario) [7]. Una característica importante del SMTP es que cada servidor físico podría operar

tanto como cliente o como servidor SMTP; al ser un servidor, acepta un correo electrónico entrante y se convierte en un cliente para reenviarlo [11].

La Capa de Conexión Segura (SSL) denota el protocolo predominante de seguridad de comunicación de Internet, en particular para los servicios World Wide Web (WWW) relacionados con el comercio electrónico o la banca hogareña. La mayoría de los servidores web y navegadores soportan SSL como el estándar de facto para la comunicación segura entre el cliente y el servidor. El protocolo SSL crea conexiones de punto a punto que permiten el uso privado y sin interrupciones intercambiando mensajes entre partes fuertemente autenticadas [12].

Actualmente, el gran crecimiento del Internet en el mundo ha penetrado en poco más del 54% de la población mundial [13], permitiendo con ello transmitir información y facilitando la comunicación entre humanos, máquinas y objetos. Esto último es el objetivo del Internet de las Cosas (IoT, por sus siglas en inglés) que versa sobre la integración del Internet con sensores, identificadores de radio frecuencia u otros objetos, en sistemas embebidos.

Los objetos dentro del IoT tienen la capacidad de enviar y recibir información, permitiéndoles comunicarse con otros dispositivos o servicios en el Internet para cumplir ciertos objetivos. De manera concreta, el IoT se define como "Una red mundial de objetos interconectados con dirección única, basada en protocolos estándar de comunicación" [14]. De hecho, el vocablo " Internet de las cosas " se atribuye a The Auto-ID Labs [15], una red mundial de laboratorios de investigación académica en el campo de la RFID (Radio Frecuency Identification) en red y las tecnologías de detección emergente.

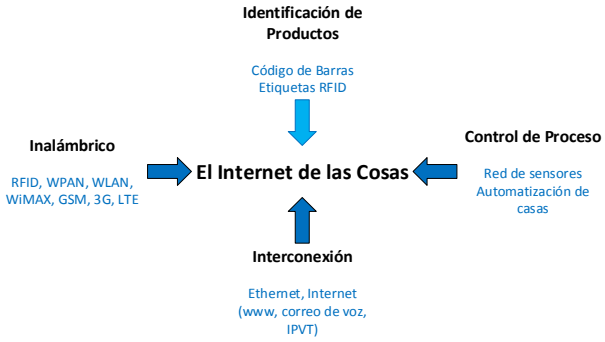

Fig. 1. Interacción de distintos componentes con el IoT [16].

En la Fig. 1, el IoT hace uso de productos de identificación, objetos inalámbricos, interconectividad y procesos de control, por tal motivo se requiere la coordinación de una entidad de mayor jerarquía que supervise el desarrollo de estas tecnologías. El uso y su consecuente importancia del IoT es de tal magnitud que, tan sólo en 2010, el número de objetos conectados en el Internet sobrepasó a la población humana [17]. Sin embargo, esto no quiere decir que el IoT sea perfecto pues aún tiene retos por afrontar. Estos se pueden resumir en los siguientes nueve puntos [18]:

- Disponibilidad de internet gratuito en todas partes.
- Temas de seguridad.

- Desarrollo de sistemas de detección inteligentes de bajo costo.
- Uso de distintos tipos de energía.
- Capacidad computacional.
- Escalabilidad.
- Tolerancia a fallos.
- Consumo de energía.
- Aceptabilidad entre la sociedad.

Además, la fragmentación de las tecnologías de la comunicación se ha considerado como uno de los principales obstáculos que entorpecen la visión de IoT. Por tanto y como requisito necesario, se tiene que el IoT debe basarse en tecnologías de comunicación estándar abiertas que permitan la interoperabilidad y las capacidades plug-and-play [19]. Esto ha motivado distintas propuestas y aplicaciones [1, 20, 21, 22] que toman ventaja de las redes sociales o de sus APIs (Interfaz de Programación de Aplicaciones) de web abierta con la intención de compartir Smart-things entre usuarios de confianza o bien, de aquellos usuarios que han aportado mecanismos interesantes y útiles destinados a integrar redes aisladas basadas en IoT en Internet a través de servicios en la nube.

El presente artículo se estructura de la siguiente forma: se aborda el estado del arte en la Sección presente, describiendo el internet de las cosas y de los correos electrónicos. La Sección 2 describe las necesidades que motivaron la creación de Ernest, los algoritmos utilizados y la arquitectura de la aplicación y, finalmente, se muestran los resultados en la Sección 3.

2. Desarrollo

Ernest es una aplicación que permite el control del encendido o apagado de las cosas —televisiones, tostadores, licuadoras, lamparas, etc.— conectadas a sus tomacorrientes haciendo uso de los correos electrónicos; al estar fundamentado en e-mails, la privacidad se mantiene, sin la intervención de terceros para acceder a la nube. Esto es posible gracias a su circuito y sus algoritmos para envío y lectura de correos electrónicos.

Actualmente, se tiene la segunda versión de Ernest, las principales diferencias entre la primera versión y la segunda, es el uso del módulo WI-FI que se emplea —un RN 131-G para la primera y un RN 1810 para la segunda—y la implementación del cifrado SSL, donde en la primera versión no se podían utilizar correos electrónicos de Gmail o Hotmail, entre otros. En este apartado se enfatizará la segunda versión por lo que se describe a continuación:

2.1. Arquitectura

Ernest cuenta con un elemento de Hardware. Dicha sección permite conocer qué enchufes están activos y cuáles no y, en caso de solicitarlo, cambiar el estado en que se encuentren. Como puede observarse en la Fig. 2, un microcontrolador ATMEGA328p tiene comunicación con un módulo RN 1810 el cuál otorga Internet al sistema embebido, también cuenta con una LCD donde se muestra el estado en el que se

encuentra el circuito, ya sea leyendo los correos, indicando si está o no conectado a Internet, o si listo en caso de haber realizado lo solicitado; y cuenta con su etapa de potencia para habilitar o deshabilitar los tomacorrientes con el uso de MOC3041.

Para energizar a Ernest se utiliza una clavija común y se conecta a un tomacorriente; para proporcionar los 5 volts requeridos por ciertos componentes del circuito, se utiliza un EME05US05, que otorga una salida de 5 V a 1 A para una entrada de 85 a 264 VAC.

Para la conexión al WI-FI se utiliza el módulo RN1810 que otorga Internet al sistema embebido y que permite la encriptación de datos con cifrado SSL a través de hardware para que no circulen sin protección por la Web.

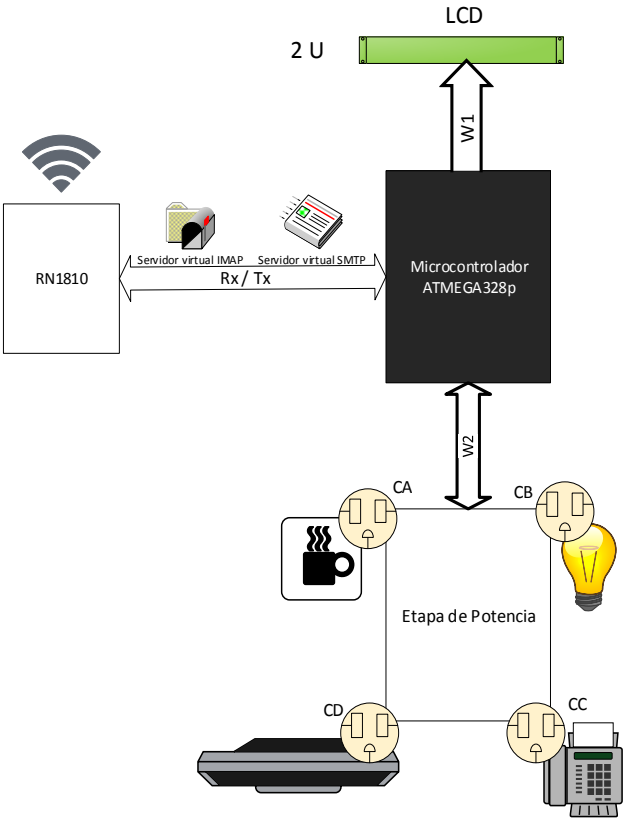

Fig. 2. Esquema del funcionamiento de Hardware de Ernest

Como medida de protección del circuito se utiliza un filtro de red, ver Fig. 3, para C.A. para eliminar parásitos en la red en modo común y diferencial; con ello se blindan los aparatos que se conecten a Ernest y a la fuente EME05US05.

Antes de pasar el circuito a una PCB, primero se prueba su funcionamiento en protoboard (Fig. 4ª). Una vez que se comprueba, se procede a pasar a la PCB, en la Fig. 4b se puede observar el circuito de Ernest en 3D donde ya se encuentra el microcontrolador, el módulo WI-FI, la etapa de potencia y el filtro, todo en el PCB.

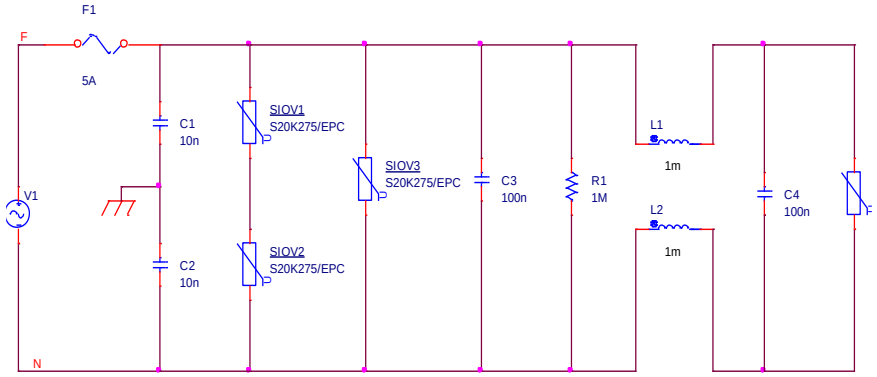

Fig. 3. Filtro de C.A. para Ernest.

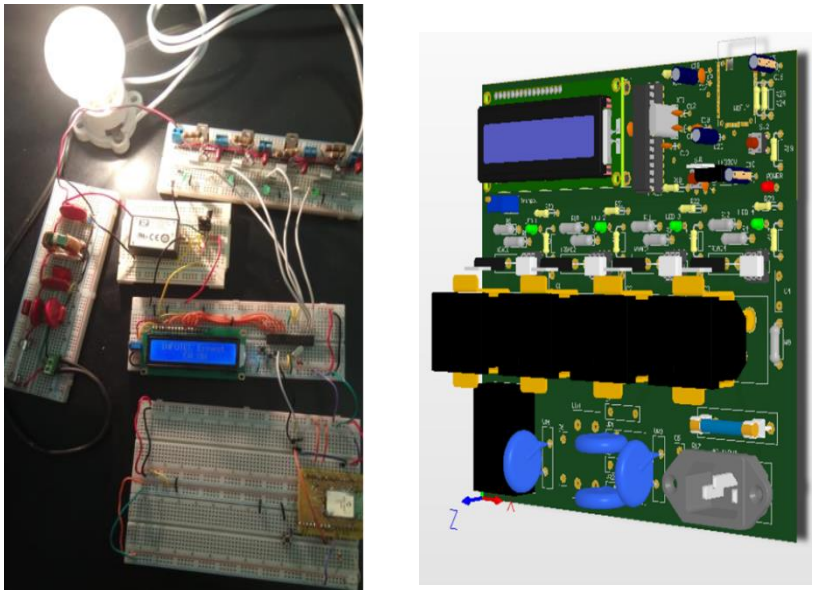

Fig. 4. a. Prueba del circuito en protoboard. b. Circuito de Ernest en 3D.

2.2. Algoritmos

Para la aplicación de Ernest se crearon tres algoritmos. Para mayor comprensión de su funcionamiento, la Fig. 6. muestra cómo Ernest hace uso del correo electrónico para la recepción de las órdenes del usuario.

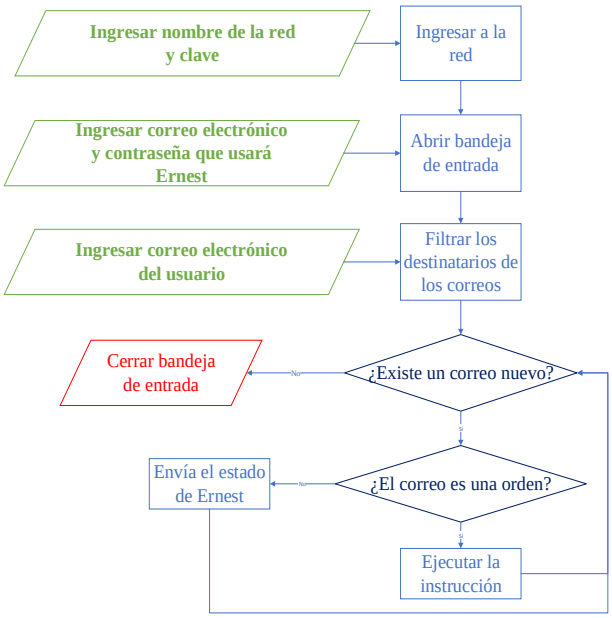

Fig. 6. Funcionamiento general de la aplicación Ernest.

Conexión al módulo Wi-Fi. Inicialmente, se comienza agregando la conectividad WI-FI al sistema con el uso del módulo Wi-Fi RN1810, utilizando un microcontrolador ATMEGA32 y generando el primer algoritmo para conectarse a la red que se desee. Tal algoritmo se esquematiza en la Fig. 7, donde se exhiben la secuencialidad a seguir para la correcta comunicación y conexión a Internet.

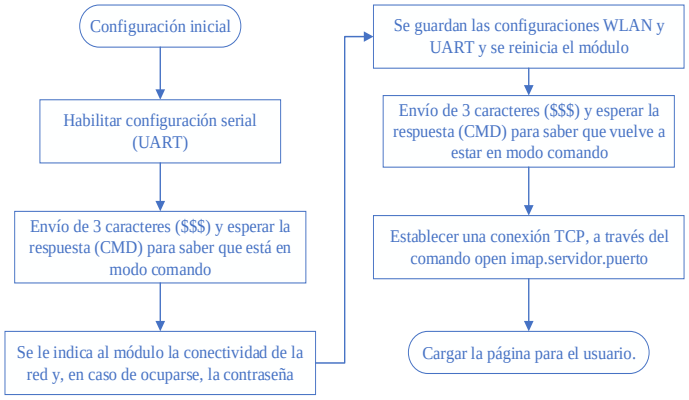

Fig. 7. Algoritmo para conectar el módulo a Internet.

Lectura de correos electrónicos. Una vez realizada la conectividad a Internet, y dado que se imprime en la LCD "Already joined network", se prosigue con el acceso al correo electrónico, haciendo uso de los protocolos de los mecanismos de recuperación

IMAP y accediendo a los mensajes de correo electrónico alojados en el servidor de éste. Tal como establecen los protocolos, se abre el IMAP por el puerto 993 del TCP (Protocolo de Control de Transmisión), mediante el comando "open IMAP 993"; una vez establecida la conexión, se saluda al servidor y se procede con la autentificación bajo el comando "logi".

Confirmado el acceso, se procede a la lectura de la bandeja de entrada. Para la lectura de los correos electrónicos nuevos, se hace uso del comando SELECT INBOX, y se utiliza después el comando FETCH seguido del número de correo por leer, tal y como estén ordenados en la bandeja. Si el correo contiene la información "prende" o "apaga", se sabe que es una instrucción que debe ejecutarse y se realiza la búsqueda de si el contacto desea "prender" o "apagar", y en caso de que no sea una instrucción, se interpreta como el estado en que se encuentra Ernest, propiciando el cambio del algoritmo de escritura de mensaje y cerrando el protocolo IMAP con el comando LOGOUT y, de la misma forma, lo cierra en caso de no haber nuevos correos sin leer.

Escritura de correos electrónicos. En la escritura del mensaje, se realiza un cambio de protocolo para pasar del IMAP al SMTP; para este protocolo se utiliza el puerto 465 TCP, iniciando con el comando "open smtp.gmail.com 465" donde gmail.com es el servidor de mensajería usado para Ernest. Para inicializar la sesión se escribe el comando "EHLO[1] Ernest", seguido de "AUTH LOGIN" y después se envía el correo electrónico y la contraseña del usuario codificado en base64. Realizado esto, se pueden mandar mensajes al HOST a través del comando MAIL FROM, el cual debe contener la dirección del remitente del sobre y se espera la respuesta. Para referenciar a quién se desea enviar el sobre, se utiliza el comando RCPT seguido de la(s) dirección(es) a la(s) que se enviará el correo. Una vez realizado esto, se puede enviar el (los) correo(s) electrónico(s), con la siguiente estructura:

DATA

From:

Subject: Ernest App

To:

Y se remite el estado de los contactos, en los cuáles se dice si los contactos en CA, CB, CC y CD están activos o no, enviándose la información de uno por uno. Una vez que se envió el mensaje, se procede a salir de este protocolo por medio del comando QUIT.

3. Resultados

En la Fig. 8 se aprecia la primera versión de Ernest donde se puede observar en funcionamiento, al estar leyendo un e-mail. En la Tabla 1 se muestra una comparativa entre ambas versiones.

[1] EHLO = (hello extendido) la respuesta del servidor a EHLO da el nombre del servidor en la primera línea.

Fig. 8. Circuito de la primera versión de Ernest

Tabla. 1. Comparativa entre las dos versiones de Ernest.

	MCU	Transceptor	Encriptación	IPv6
Ernest 1ra versión	Atmega328p	Wifly rn-131G	NO	NO
Ernest 2da versión	Atmega328p	Wifly rn-1810	SSL	SI

Con la segunda versión de Ernest se desarrollaron pruebas de fidelidad. Antes de enviar algún correo se verificó que Ernest estuviera estableciendo una conexión exitosa con la red WI-FI y con el servidor IMAP. La prueba consistió en dejar conectada la placa por una hora revisando la transmisión a través del puerto serie, véase Fig. Cuando el internet se encontraba saturado o lento, o bien, si el módulo WI-FI tenía una señal muy baja -observado a través de dispositivos cercanos al área de trabajo de Ernest- se obtuvieron fallas con la conexión del servidor IMAP.

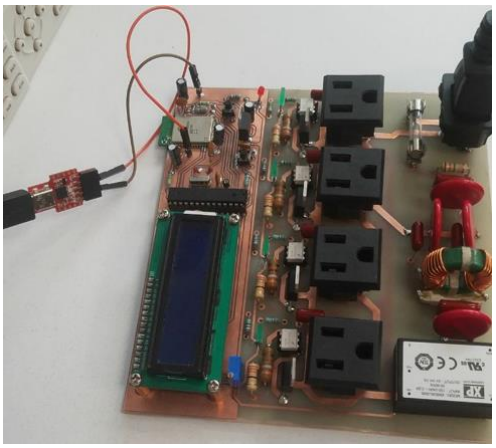

Fig. 9. Conexión de prueba del microcontrolador con el módulo Wifi.

Para comprobar la eficacia de dichos algoritmos, se realizaron dos pruebas en las cuales durante una hora se enviaron primero 50 e-mails y después en una segunda prueba se mandaron 100 e-mails y se comprobó que estos fuesen recibidos por Ernest, y de la misma forma, que fuesen ejecutado por este. Durante la primera prueba todos los correos enviados fueron recibidos. En la segunda prueba se enviaron 100 correos, en el lapso de esta hora se volvió a observar lo que ocurría en el puerto serie. Los primeros 52 correos que leyó se ejecutaron correctamente, el correo número 53 no lo ejecutó debido a que el Wifly obtuvo una conexión fallida con el servidor IMAP. Las posibles causas de esto son las siguientes:

(1) Se perdió la conexión WI-FI.
(2) El servidor de IMAP rechazó la solicitud.

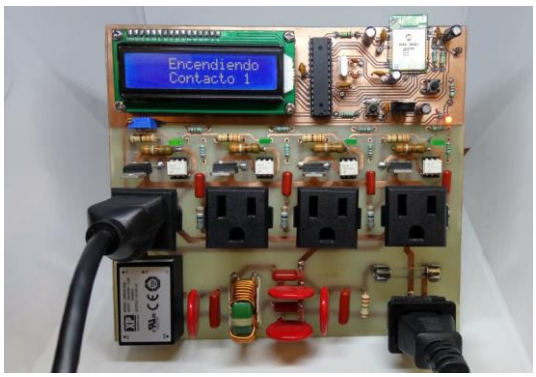

Fig. 10. Ernest ejecutando comandos.

Una vez que se reinició el módulo WI-FI, Ernest continuó su trabajo de lectura de correos normalmente, del 54 al 100. En la Tabla 2 se muestra la efectividad de Ernest

en ambas pruebas con solo uno correo no recibido durante la segunda prueba. En la Fig. 10 se muestra a Ernest durante las pruebas realizadas en la recepción de correos.

Tabla. 2. Efectividad de Ernest para recibir correos.

	Correos enviados	**Correos exitosos**	**Porcentaje**
Prueba 1	50	50	100.00%
Prueba 2	100	99	99.00%

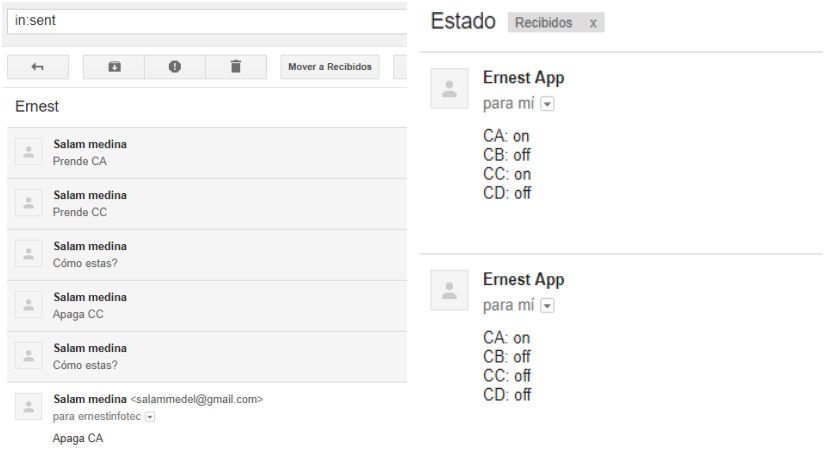

Fig. 11. a. E-mails enviados a Ernest b. E-mails recibidos de Ernest.

En la Fig. 11a puede observarse un ejemplo en el que se enviaron seis e-mails a Ernest de los cuáles cuatro fueron órdenes para su ejecución y los otros dos fueron preguntas para conocer el estado en que se encuentra. Las respuestas recibidas se observan en la Fig. 11b, donde se confirma que se ha realizado exitosamente el encendido o el apagado de un contacto.

4. Conclusiones

A largo del presente trabajo se presentó un sistema embebido para el control de casas inteligentes mediante correos electrónicos con cifrado SSL. Así como su metodología de desarrollo acompañada de una serie de pruebas para validar el funcionamiento del sistema. Ernest representa una opción tangible para aplicaciones o bien usuarios sensibles a compartir su información personal en una nube. Durante la experimentación, se demostró que Ernest tiene una certeza del 99 % de eficacia, lo cual demostró ser un prototipo funcional y confiable. Sin embargo, para poder tener una mayor certeza con respecto a las posibles fallas aún deben de realizarse más pruebas en distintos ambientes para poder obtener datos ya sea de cómo influye el tráfico de la red en la conectividad del módulo. Además, durante los experimentos realizados se observó

un gran desempeño por parte de los algoritmos para ejecutar las instrucciones, obteniendo más dificultades en el apartado de conexión al módulo Wi-Fi

Ernest también presenta áreas de oportunidad para ser un sistema más robusto. Actualmente, se trabaja en el desarrollo de la próxima versión de Ernest, donde se empieza a trabajar con el procesamiento de textos con la finalidad que Ernest se vuelva más intuitivo a las demandas de sus usuarios. También se está trabajando en un proceso handshake para evitar que ese bajo porcentaje de correos no ejecutados se corrija cuando al no recibir respuesta del sistema el mismo correo sea retransmitido. Por último, se observó que se debe cambiar de microcontrolador a causa de la poca memoria RAM que contiene el ATMEGA328p, por ello se está pasando a un STM32F42 por su capacidad de 192 Kbyte de la misma, permitiendo cargar con buena calidad la página del servidor del correo electrónico, además de que se le realizará su respectiva placa, pero ahora de montaje superficial.

Referencias

1. SmartThings–Make your world smarter, http://smartthings.com/, last accessed 2018/03/13
2. Atzori, L., Iera, A., Morabito, G.: The Internet of Things: A survey. Comput. Netw., pp. 2787–2805 (2010)
3. Alhakbani, N., Hassan, M.M., Hossain, M.A., Alnuem, M.: A Framework of Adaptive Interaction Support in Cloud-Based Internet of Things (IoT) Environment. In: Fortino G., Di Fatta G., Li W., Ochoa S., Cuzzocrea A., Pathan M. (eds) Internet and Distributed Computing Systems. Lecture Notes in Computer Science, vol. 8729, pp. 136–146. Springer, Cham (2014)
4. Joyanes-Aguilar, L.: Computación en la Nube. 2nd edn. Alfaomega, Panamá (2012)
5. Guerrero-Ibáñez, J. A., Flores-Cortés, C., Zeadally, S.: Vehicular Adhoc Networks (VANETs): Architecture, Protocols and Applications. In: Next-Generation Wireless Technologies, pp. 49–70. Springer, London (2013)
6. Stankovic, J. A.: Research Directions for the Internet of Things. In: Life Fellow, Vol. 1, No. 1, pp. 3–9. IEEE Internet of Things Journal, Charlottesville (2014)
7. Meinel, C., Sack, H.: Application Layer and Internet Applications. In: Internetworking, pp. 665–830. Springer, Berlin (2013)
8. Klensin, J.: Simple Mail Transfer Protocol, IETF RFC 5321 (2008)
9. Myers, J., Rosem, M.: Post Office Protocol – Version 3, IETF RFC 1939 (1996)
10. Crispin, M.: Internet Message Access Protocol – Version 4rev1, IETF RFC 3501 (2003)
11. Pakulin, N., Tugaenko, A.: Specification Based Conformance Testing for E-mail Protocols, pp. 371–381. Springer, Berlin (2010)
12. Tilborg H. C. A., Jajodia S.: Encyclopedia of Cryptography and Security. 1st edn. Springer, Boston, MA. (2011)
13. Internet World Status - Usage and Population Statistic http://www.internetworldstats.com/stats.htm (2018)
14. INFSO D.4 Networked Enterprise & RFID INFSO G.2 Micro & Nanosystems, in: Co-operation with the Working Group RFID of the ETP EPOSS, Internet of Things in 2020, Roadmap for the Future, Version 1.1 (2008)
15. Auto-Id Labs (2018)
16. Mainwaring, K., Srivastava, L.: The Internet of Things. In: H. Chaouchi (Ed.), ISTE edition (2010)

17. Evans, D.: The internet of things: How the next evolution of the internet is changing everything, CISCO White Paper (2011)
18. Mukhopadhyay, S. C., Suryadevara, N. K.: Internet of Things: Challenge and Opportunities. volume 9. In: Springer (2014)
19. Colitti, W., Long, N. T., De Caro, N., Steenhaut, K.: Embedded Web Technologies for the Internet of Things, Chandra Mukhopadhyay. In: Springer (2014)
20. Guinard, D., Fischer, M., Trifa, V.: Sharing using social networks in a composable Web of Things. In: 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 702–707 (2010)
21. Cosm–Internet of Things Platform Connecting Devices and Applications for Real–Time Control and Data Storage (2013) (2018)
22. Paraimpu–The Web of Things is more than Things in the Web (2018)