El Monitoreo de Humedad y Temperatura de un Centro de Datos

Rodolfo Omar Domínguez García, PhD[1], Omar Ali Zatarain Duran, PhD[1,2], Miguel Ángel de la Torre, PhD[1], José Israel Díaz Vieyra and Diego de Jesús Quijas Santos.
[1]Universidad de Guadalajara, México, odomi@academicos.udg.mx, miguel.dgomora@academicos.udg.mx,
[2]Universidad de Guadalajara, México, omar.zatarainduran@ucalgary.ca, diego_quijassantos@hotmail.com, Universidad Politécnica del Valle de Toluca, *isra_695@hotmail.com*

Abstract*– El monitoreo y control de humedad y temperatura de un Centro de Datos (DC – Data Center) ha representado un reto para su operación ininterrumpida, debido a fluctuaciones en el flujo de datos y condiciones ambientales. Diferentes tecnologías agrupadas en el llamado Internet de las Cosas (IoT) permite el monitoreo de cualquier dispositivo en forma remota sin restricción de la heterogeneidad de los equipos. Por un lado, las implementaciones que hay son cerradas y de alto costo, además las distintas soluciones propuestas en la literatura utilizan plataformas con restricciones en su configuración, debido a su carácter de licenciamiento. Particularmente en el ámbito educativo, el uso de plataformas abiertas y de bajo costo propicia su utilización en el aula, promoviendo el aprendizaje y posterior desarrollo de este tipo de tecnología. En este artículo se presenta la descripción de la implementación del sistema de monitoreo remoto de humedad y temperatura en el DC principal del Centro Universitario de los Valles (CUValles) de la Universidad de Guadalajara. Este sistema está implementado utilizando una tarjeta Raspberry Pi 3B+, bajo Raspbian v14.3, usando un sensor DHT11 y un puerto Ethernet. La programación se ha desarrollado utilizando Python 3.x, con funciones que permite, recabar la información del sensor DHT11, guardarla en la base de datos sqlite de la propia raspberry y transmitirla a la base de datos de la nube del CUValles. La BD está implementado sobre Ubuntu, con MySql como gestor de bases de datos, y se desarrolló una APP de monitoreo y alertas bajo Android, que permite la oportuna atención de contingencias.

***Keywords*—Data Center, Raspberry Pi, Python, DHT11 e IoT.**

I. Introduction

El uso del internet y el fácil acceso a la información que se tiene hoy en día, han contribuido a materializar la idea de controlar todos los dispositivos que nos rodean. Esto sugiere la operación y monitoreo de equipos de manera remota, en que el operador asume una postura omnipresente. La tendencia actual se refiere al uso de la computación ubicua, y el concepto de internet de las cosas (IoT) que se han constituido como el conjunto de tecnologías del futuro, que favorecen el desarrollo de escenarios en que los dispositivos están totalmente conectados. De acuerdo con los datos proporcionados por Cisco se calcula que en el 2020 sea de uso común esta metodología de integración, contribuyendo a la revolución del internet que pretende mantener todo interconectado, y que actualmente es utilizado en pequeñas magnitudes y por separado [1]. Un ejemplo son los automóviles, que utilizan múltiples redes para el manejo del motor y monitoreo del rendimiento y operación. De manera similar, los edificios y residencias inteligentes tienen distintos sistemas de control para la seguridad, iluminación, calefacción, etc. Las nuevas tecnologías están incorporando herramientas aún más poderosas, con el afán de implementar dispositivos capaces de cooperar entre sí, para obtener información de éstos de manera remota, y tener control de estos dispositivos. Esto ha sido posible gracias a las interfaces que brindan acceso a la web mediante información proveniente de diversas fuentes interconectadas a través de internet, ubicadas en diversos puntos de acceso y con múltiples dispositivos capaces de conectarse. La vigilancia se realiza mediante la ejecución de aplicaciones móviles o a través de páginas web que permiten visualizar múltiples eventos de manera gráfica y cuantificable. La reducción de tamaño y precios de los sensores y actuadores disponibles en el mercado, hacen cada día más práctica y estética su implementación en prototipos experimentales, mejorando características tanto en hardware como en software. Al final, la incorporación de estos dispositivos permite desarrollar controles más complejos y eficientes en el momento de dotar de internet a los elementos a conectar, permitiendo tanto el monitoreo como la manipulación remota del hardware. Esto dio paso al nacimiento, no hace mucho tiempo, de dispositivos que consisten en tarjetas electrónicas para el desarrollo de sistemas embebidos, facilitando aún más la tarea de integrar sensores y controladores vía estos dispositivos electrónicos con "inteligencia", se tienen varios ejemplos muy exitosos como son las tarjetas: Arduino, Samsung ARTIK 710 y Raspberry Pi, entre otras

II. Tendencias en Internet de las Cosas (IoT)

Kevin Ashton fue pionero en la proposición del concepto de IoT en 1999, y él se refería a la interconexión de objetos con identificación única que podían interoperar con una tecnología de identificación única de radio frecuencia (RFID) por sus siglas en inglés. Sin embargo, la definición más apropiada de IoT está en proceso de formación ya que está sujeta a las perspectivas que se tomen [2].

IoT ha sido definida como la "infraestructura global de redes dinámicas con capacidades de auto-configuración basados en estándares y protocolos de comunicación interoperables; cosas físicas y virtuales, en una IoT se tiene identidades y atributos capaces de usar interfaces inteligentes estando integrados como una red de información" [2].

Digital Object Identifier: (to be inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).

Básicamente, el IoT puede ser tratado como un súper conjunto de dispositivos de conexión que son identificables de manera única por las técnicas existentes de comunicación en campo cercano por sus siglas en inglés (NFC) [2].

La International Telecommunication Union (ITU) discutió la habilitación de tecnologías, mercados potenciales y retos emergentes y la implicación de la IoT [3]. La evolución de IoT puede ilustrarse por varias fases como se muestra en la figura 1.

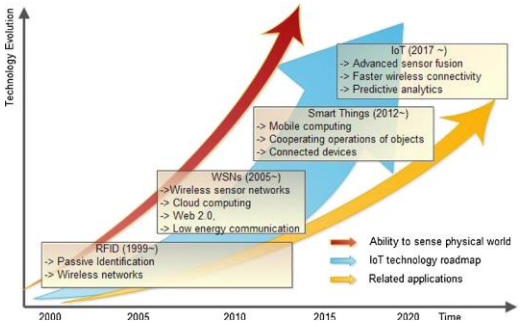

Fig.1 Evolución de la Tecnología. [2]

La IoT se inicia mediante el uso de la tecnología RFID, que se utiliza cada vez más en logística, producción farmacéutica, entre otras. Debido a las redes de sensores inalámbricos por sus siglas en inglés, (Wireless sensor networks, WSN), también llamadas redes de sensores y actuadores (Wireless sensor and actuator networks, WSAN) son sensores autónomos espacialmente distribuidos para monitorizar condiciones físicas o ambientales, como temperatura, sonido, $C0_2$, entre otros [4]. Lo que ha ampliado significativamente las capacidades de sensado de los dispositivos y, por lo tanto, el concepto original de IoT se extiende a la inteligencia ambiental y al control autónomo. Hasta la fecha, varias tecnologías están involucradas en IoT, como las WSN, códigos de barras, detección inteligente, RFID, NFC, comunicaciones inalámbricas de bajo consumo como Xbee, computación en la nube, y así sucesivamente. La evolución de estas tecnologías traen nuevas herramientas para IoT [2]. La evolución del IoT impulsa la creación de proyectos de investigación y desarrollo tanto en la academia como en la industria.

En este caso y debido al surgimiento de la pequeña computadora de bajo costo "Raspberry Pi", que ha tenido una gran aceptación en aplicaciones como la automatización, monitoreo y control. Aun cuando originalmente fue pensada para cuestiones de educación y aprendizaje de la informática, se ha adaptado muy bien en lo que hoy se llama el IoT, o una de sus áreas llamada Internet Industrial de las Cosas (IIoT) por su facilidad de implementación, bajo costo y gran precisión.

III. Monitoreo del DC de CUValles

A. *Centro de Datos Monitoreado en CUValles*

El compromiso que se tiene de contar con infraestructura tecnológica que opere y esté disponible las 24 horas del día, los 365 días del año, en los Centros de Datos o Data Center (DC), siendo los espacios neurológicos por excelencia de la información, obliga a contar con herramientas tecnológicas que vigilen las principales variables que impactan en la operación ininterrumpidas de estos DC, dos de las cuales son la temperatura (T) ambiental para evitar principalmente el sobrecalentamiento de los equipos y la humedad relativa (HR), ya que normalmente se dice que la HR debe estar entre el 40% y el 55%, ¿por qué?, porque por encima del 55% aumentaríamos el riesgo de corrosión, y por debajo del 40% aumentamos el riesgo de descargas estáticas[5].

Podemos observar en la figura 2, las especificaciones que deben de establecerse en un Data Center.

ASHRAE
Advancing HVAC&R to serve humanity and promote a sustainable world

Datacom Equipment Environment Specifications

Class	Allowable Temp	Recommended Temp	Allowable Relative Humidity	Recommended Relative Humidity	Max Rate of Change
1	15-32° C 59-90° F	20-25° C 68-77° F	20-80%	40-55%	5° C 10° F

Fig. 2 Especificaciones del medio ambiente de un Data Center[5]

Existen dos posibles amenazas relacionadas con la humedad relativa dentro del Data Center:

- **Descargas electroestáticas:** las posibilidades de descargas electroestáticas, también conocidas como ESD (electrostatic discharge) se producen cuando la humedad baja. Asimismo, esas posibilidades aumentan aún más si la temperatura es baja. Las descargas electroestáticas pueden ser apenas perceptibles para las personas, pero no causan ningún tipo de daño. En cambio, una descarga de 10 Volts, ya es capaz de dañar un equipo.
- **Corrosión:** ocurre cuando un elemento metálico es expuesto al agua, ya sea porque se moja o se generan pequeñas gotas causadas por la condensación de agua en el aire. Por ejemplo: en un ambiente con una humedad alta. Los elementos dentro de los servidores se pueden dañar y sufrir una pérdida de datos.

La clave es encontrar un equilibrio justo para tener la humedad en un rango óptimo donde se eviten las descargas electroestáticas y de condensación. Por ello, el rango más adecuado de humedad es entre el 40% y el 55% (también es el rango recomendado por la norma TIA/EIA 942) [5].

B. *Diseño*

El esfuerzo realizado por estudiantes y profesores del CUValles de la Universidad de Guadalajara para impulsar el

desarrollo e incorporación de tecnologías IoT a la vida de la región, se ha visto reflejado en múltiples proyectos. Entre ellos, está la implementación del proyecto 'Monitoreo de Centro de Datos', en el cual se emplean las herramientas descritas a continuación.

Tarjeta Raspberry Pi 3B+ (Pi)

La tarjeta Raspberry Pi es una computadora pequeña y fácil de usar, basada principalmente en Raspbian SO de Linux distribución Debian, usada entre otras cosas para desarrollar sistemas embebidos [6]. Es un ordenador del tamaño aproximado de una tarjeta de crédito. Consta de una placa base sobre la que se monta un procesador, un chip gráfico y memoria RAM [6]. Sus principales componentes son: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit quad-core SoC @ 1.4GHz, 1GB LPDDR2 SDRAM, ver figura 3. Posee varios puertos entre ellos un Puertos Ethernet de 100Mbs [7].

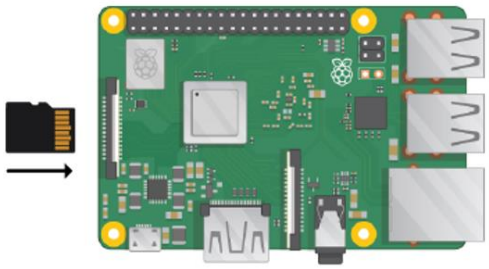

Figura 3. Raspberry Pi 3B+[7]

Sensor DHT11

El sensor DHT11 es comúnmente utilizado para monitorear tanto temperatura como humedad, y su operación es completamente digital, ver figura 4. Es decir, no requiere una compuerta analógica para su lectura. Su rango de temperatura es de 0 - 60 ºC, con un rango de error de ±, 1º, su rango de humedad relativa de 0 – 60%, con un rango de error de +-5% y tensión de funcionamiento 5V [8].

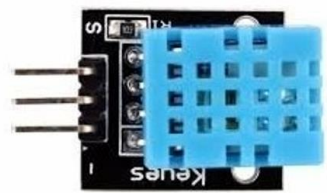

Figura 4. Sensor de humedad y temperatura, DHT11 [7].

La plataforma de desarrollo de software para el DC está basada en uno de los lenguajes de programación más populares entre los desarrolladores de sistemas para análisis de datos: Python. Python es un lenguaje de programación poderoso, cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos, haciéndolo ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas y sistemas embebidos [9].

El sistema gestor de bases de datos ligero utilizado en el desarrollo es Sqlite, que es una biblioteca escrita en leguaje C que implementa un Sistema de gestión de bases de datos transaccionales SQL autocontenido, sin servidor y sin configuración. El código de Sqlite es de dominio público y libre para cualquier uso, ya sea comercial o privado [10]. Las características y plataformas previamente mencionadas hacen de SQLite una excelente opción en diversos casos tales como:

- Desarrollo de aplicaciones en dispositivos móviles que manejan una base de datos local que se sincroniza por batch con una base de datos remota [11].
- SQLite es una Base de datos propia para el Internet de las cosas. Es una opción popular para el motor de base de datos en teléfonos móviles, PDA, otros dispositivos electrónicos. Hace un uso eficiente de la memoria, el espacio en disco y el ancho de banda del disco, es altamente confiable y no requiere mantenimiento de un administrador de base de datos [12].

Por otro lado, se utiliza el sistema gestor de bases de datos MySQL que es la base de datos de código abierto más popular del mundo por ofrecer un alto rendimiento confiable y rentable y comercio electrónico escalable, procesamiento de transacciones en línea y base de datos, integrada a aplicaciones [13].

C. Desarrollo

Para el desarrollo de esta aplicación se inició configurando la Raspberry pi 3B+ para eso se le instaló el SO Raspbian en su última versión 4.14, El uso del software NOOBS es la forma más fácil de instalar Raspbian en su tarjeta SD. NOOBS es un instalador de sistema operativo que contiene Raspbian [14]. Una vez que se tiene la imagen del SO, se descomprime y se copia a una tarjeta SD de 16GB vía una computadora portátil, previamente formateada con la herramienta SD Card Formatter [15], se instala la microSD en la Raspberry Pi.

Una vez montada SD en la Raspberry se procede a conectarle todos los dispositivos necesarios incluyendo una pantalla por el puerto HDMI, así como su respectivo teclado y mouse y la fuente de poder 5V., 2,5A, para su arranque. Una vez que se inicializa obtenemos un escritorio Linux de Raspbian, se termina de configurar y seguidamente de eso se aplicaron los siguientes comandos para la actualización del sistema:

$sudo apt-get update
$sudo apt-get -y upgrade

A continuación, se procedió a conectarle el sensor DHT11, como se observa en la figura 5, y Se descargó e instaló la librería de DHTxx con los siguientes comandos:

$git clone https://github.com/jdupl/dhtxx-rpi-python3
$Sudo python3 setup.py install

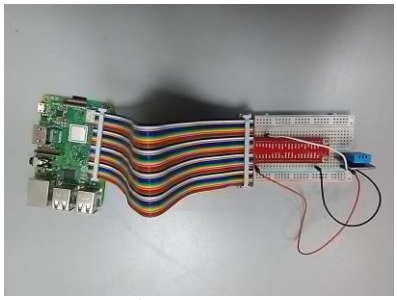

Figura 5. Conexión del DHTII a la Raspberry Pi

Dentro del escritorio de Raspbian se tiene el programa Thonny Python IDE, que maneja la versión Python 3, el cual fue el que se usó para generar los programas de la aplicación, en el siguiente código se probó la lectura del sensor DTH11 que se muestra en la Caja 1.

```
#importando librerías a utilizar
form time import sleep
form dhtxx import DHT11
#se declara el puerto donde está conectado el
sensor
dht11=DHT11(14)
#inicio del bucle
While True:
   #intentos a realizar para obtener resultados
validos
   r = dht11.get_result(max_tries=10)
   if r:
#se muestran los resultados almacenados en el
buffer r
      print('Temp: {0:0.1f} C Humidity; {1:0.1f}
%'.format(r[0], r[1]))
   else:
      print('Failed to get result!')
   sleep(1)
```

Caja 1. Lectura del sensor

Al aplicar el código de lectura del sensor se obtienen las siguientes lecturas:

temp: 25.2, hum:62%
temp: 25.3, hum:60%
temp: 25.0, hum:58%
temp: 25.6, hum:59%
temp: 25.6, hum:59%

que son las lecturas esperadas y que están en rango del lugar donde se está midiendo (laboratorio de pruebas).

Creación de base de datos (BD)

El esquema de la base de datos es la que se observa en la figura 6, que es la que se diseñó e implementó. Como primer paso se inició la creación de la base de datos en SQLite, ya que se toma como una librería integrada a Python. En la caja 2 se muestra el código que se utilizó para crear la base de datos, este código además de crear la base de datos también funciona para verificar si la tabla existe, si es así la elimina con todo y los datos contenidos. Enseguida crea la tabla requerida con los campos especificados finalizando al cerrar la conexión a la base de datos.

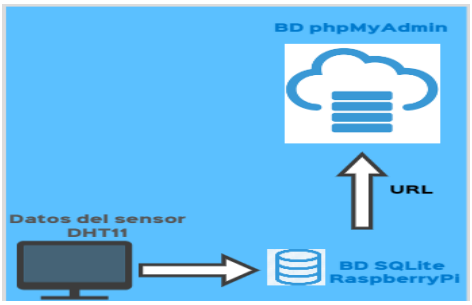

Figura 6. Conexión de las BD

```
#se importa la librería SQLite.
Import sqlite3
#se crea la BD como si se mandara a llamar
conn= sqlite3.connect(´cta.sqlite´)
cur.conn.cursor()
#se borran los datos de la tabla como un
reinicio.
cur.execute(´DROP TABLE IF EXISTS Site1´)
#se crea la tabla con los campos a utilizar.
cur.execute(´CREATE TABLE Site1(Id INTEGER
PRIMARY KEY AUTOINCREMENT, Fecha TEXT,
Temperatura Real, Humedad Real)´)
#cierra la conexión de la BD.
conn.close()
```

Caja 2. Creación de la base de datos

Lectura de datos.

Se muestra el diseño de la base de datos en la figura 7.

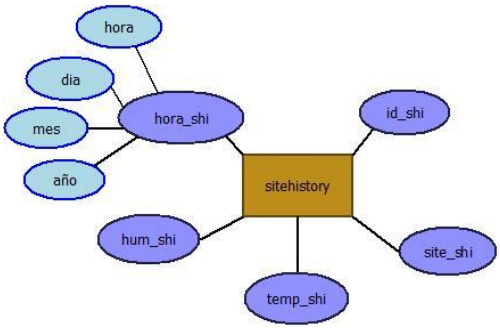

Figura 7. Esquema de la base de datos

Para la inserción de los valores enviados por el sensor a la base de datos se codificó el siguiente algoritmo, bajo los siguientes requerimientos: el primero registra la humedad y temperatura inicial, el segundo registra la humedad y temperatura cada 30 minutos en caso de que este se

```
#importando librerías a utilizar
from time import sleep
from dhtxx import DHT11
import sqlite3
import time
import datetime
#se declara el puerto donde está conectado el
sensor
dht11 = DHT11(14)
#conecta la BD
conn= sqlite3.connect('cta.sqlite')
cur= conn.cursor()
#variables utilizadas.
i=0
E=True
while True:
 #intentos a realizar para obtener resultados
válidos.
 r = dht11.get_result(max_tries=10)
 #obtención y configuración del tiempo para el
registro de la fecha y hora.
 unix= int(time.time())
 date=
str(datetime.datetime.fromtimestamp(unix).strfti
me('%Y-%m-%d %H:%M:%S'))
 if r:
  print('Temp:  {0:0.1f}  C  Humidity;  {1:0.1f}
%'.format(r[0], r[1]))
  i= i+1
  #Almacena los datos de iniciales.
  if E==True:
   cur.execute('INSERT     INTO     Site1(Fecha,
Temperatura,   Humedad)   VALUES   (?,?,?)',(date,
r[0], r[1]))
   E=False
  #rangos     establecidos     para     almacenar
elevaciones de temperatura y/o humedad.
  if r[0]>=28:
   cur.execute('INSERT     INTO     Site1(Fecha,
Temperatura,   Humedad)   VALUES   (?,?,?)',(date,
r[0], r[1]))
  elif r[1]>=80:
   cur.execute('INSERT     INTO     Site1(Fecha,
Temperatura,   Humedad)   VALUES   (?,?,?)',(date,
r[0], r[1]))
  if i== 3:
   print('30 min')
   i=0
   cur.execute('INSERT     INTO     Site1(Fecha,
Temperatura,   Humedad)   VALUES   (?,?,?)',(date,
r[0], r[1]))
 else:
   print('Failed to get result !')
 sleep(60)
 conn.commit()
cur.close()
```

Caja 3. Inserción de variables en la base de datos

mantuviera casi constante y tercero si la humedad o temperatura se elevaban o disminuyen en un rango de valores establecidos estas variaciones son almacenadas. Cada vez que se hace una inserción a la base de datos se registra la hora y fecha en que ocurre cada suceso, el código se muestra en la caja 3.

Creación de gráficas

Para la visualización de los datos de una manera más factible tal como una gráfica donde se aprecia el comportamiento de los datos en términos cuantificables, se utilizó la librería matplotlib la cual fue instalada en el SO de la Raspberry desde la consola con los siguientes comandos.

$Sudo apt-get update //busca actualizaciones.
$sudo apt-get upgrade //instala las actualizaciones encontradas.
$sudo apt-get install python3-matplotlib //instala la librería deseada.

Una vez instalada la librería se grafica los datos de temperatura con el algoritmo en Python especificado en la caja 4 y la caja 5.

```
#importando librerías a utilizar.
import sqlite3
import time
import datetime
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from dateutil import parser
from matplotlib import style
#Estilo de la gráfica.
style.use('fivethirtyeight')
#conexión de a la BD.
conn= sqlite3.connect('cta.sqlite')
cur= conn.cursor()

#función para graficar.
def graf_data():
 #consulta para extraer los datos
almacenados.
    cur.execute('SELECT Fecha, Temperatura,
Humedad FROM Site1')
    data= cur.fetchall()
    #almacena los datos en buffers.
    dates = []
    values= []
    values1=[]
    #llena los buffers conforme lee las filas
de la tabla en la BD.
    for row in data:
        dates.append(parser.parse(row[0]))
        values.append(row[2])
        values1.append(row[1])
#función para tener un conjunto de gráficas.
    fig, ax1=plt.subplots()
#configuración de la gráfica de humedad.
    color='tab:red'
ax1.set_xlabel('fecha y hora')
ax1.set_ylabel('Humedad', color=color)
```

Caja 4. Graficado de la temperatura

```
#    ax1.plot_date(dates, values, color=color)
    ax1.tick_params(axis='y',
labelcolor=color)

#crea el eje opuesto al original para
cuantificar la segunda gráfica.
    ax2=ax1.twinx()
    #configuración de la gráfica temperatura.
    color='tab:blue'
    ax2.set_ylabel('Temperatura', color=color)
    ax2.plot_date(dates, values1, '-')
    ax2.tick_params(axis='y', labelcolor=color)
    #ajusta de manera automática el rango de
valores.
    fig.tight_layout()
    #muestra la grafica.
    plt.show()
graf_data()
cur.close
conn.close()
```

Caja 5. Graficado de la temperatura (continuación)

obteniendo como resultado la siguiente gráfica, que se observa en la figura 8.

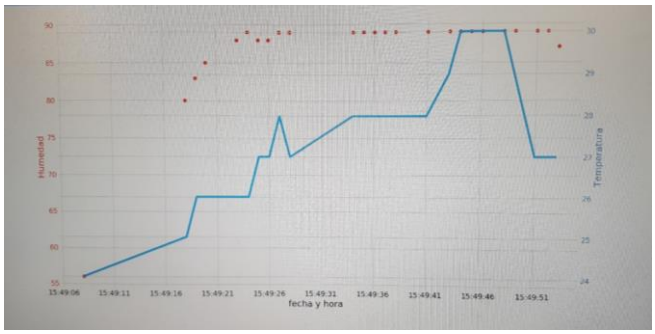

Figura 8. Gráfica de humedad y temperatura con Matplotlib

Como se puede apreciar en ella se encuentran dos trazos cada uno de color distinto (azul y rojo) y en los ejes verticales las magnitudes son del mismo color correspondiente a su trazo, las muestras se tomaron en lapsos de tiempo corto.

Almacenamiento de datos

Los datos que se están recolectando en la Base de Datos (DB) de la Raspberry Pi, deben ser almacenados de forma permanente en la DB de la nube interna del CUValles ya antes mencionada, esto se hace a través de una URL: http://148.202.89.58/site/?nom=siteuno&temp=23.0&hum=32.9

Conexión y almacenamiento en el servidor

Hasta ahora se tiene todo funcionando por partes, se requiere comunicar ambas bases de datos, esto se resolvió importando la librería urllib.request y enviando los datos bajo la lógica que se planteó con anterioridad. Como se muestra en la caja 6, se manda la URL concatenando los valores obtenidos por el sensor DHT11, de esa manera la Raspberry puede seguir guardando los valores en su DB y a la vez mandarlos a la DB de la nube, por lo que, si se llegase a perder conexión a internet o alguna falla similar se mantendría el registro de los datos en la Raspberry, una vez restablecida la conexión a internet se actualizaría los datos en la DB de la nube.

```
import urllib.request
   contents                                    =
urllib.request.urlopen("http://148.202.89.58/sit
e/?nom="+str(site)+"&temp="+str(r[0])+"&hum="+st
r(r[1])).read()
```

Caja 6. Conexión con la URL

El código final que corre en la Raspberry Pi se desarrolla en las cajas 7, 8 y 9 en el mismo orden.

```
#BD_resp.py
import  time
from  dhtxx  import  DHT11
import datetime
import sqlite3
import       urllib.request,       urllib.parse,
urllib.error
dht11=DHT11(14)
i = 0
n = 0
e=True
site='Inicio2'
temp='Temp2'
hum='Hum2'
check='Revision2'
conn= sqlite3.connect('cta.sqlite')
cur= conn.cursor()
conn2 = sqlite3.connect('respaldo.sqlite')
cur2 = conn2.cursor()
conn2.execute("CREATE   TABLE   IF   NOT   EXISTS
tabla_resp(IDr      INTEGER      PRIMARY      KEY
AUTOINCREMENT, Fecha_resp TEXT, Temperatura_resp
TEXT, Humedad_resp TEXT)")
def respaldo():
    cur2.execute('INSERT                    INTO
tabla_resp(Fecha_resp,          Temperatura_resp,
Humedad_resp)   VALUES   (?,?,?)',(date,   r[0],
r[1]))
    conn2.commit()
def mover():
    print('BD resp')
    cur2.execute("SELECT Temperatura_resp,
                Humedad_resp FROM tabla_resp")
    for row in cur2:
        cur.execute("INSERT INTO Site1
        (Temperatura, Humedad) VALUES (?,?)",
        (row))
        conn.commit()
        contents = urllib.request.urlopen(
"http://148.202.89.58/site/?nom="+str('Site_resp
')+"&temp="+str(row[0])+"&hum="+str(row[1]))
        print(row)
cur2.execute('DROP TABLE IF EXISTS tabla_resp')
```

Caja 7. Código de operación en la Raspberri Pi

```
cur2.execute("CREATE TABLE IF NOT EXISTS
   tabla_resp(IDr INTEGER PRIMARY KEY
         AUTOINCREMENT, Fecha_resp TEXT,
     Temperatura_resp TEXT, Humedad_resp
TEXT)")


def mover():
    print('BD resp')
    cur2.execute("SELECT     Temperatura_resp,
Humedad_resp FROM tabla_resp")
    for row in cur2:
        cur.execute("INSERT     INTO     Site1
(Temperatura, Humedad) VALUES (?,?)", (row))
        conn.commit()
        contents                                =
urllib.request.urlopen("http://148.202.89.58/
site/?nom="+str('Site_resp')+"&temp="+str(row
[0])+"&hum="+str(row[1]))
        print(row)
    cur2.execute('DROP    TABLE    IF    EXISTS
tabla_resp')
    cur2.execute("CREATE TABLE IF NOT EXISTS
tabla_resp(IDr     INTEGER     PRIMARY     KEY
AUTOINCREMENT,         Fecha_resp         TEXT,
Temperatura_resp TEXT, Humedad_resp
TEXT)")


def no_conex():
  while True:
    unix= int(time.time())
    date = str(datetime.datetime.
           fromtimestamp(unix).
           strftime('%Y-%m-%d %H:%M:%S'))
    global n
    if r:
       n = n +1
       print('Temp:  {0:0.1f}  C  Humidity:
              {1:0.1f}  %'.format(r[0],
r[1]))
       time.sleep(1)
       if n == 1500:
         print('No esta conectado a la BD')
         respaldo()
           try:
             if(urllib.request.urlopen(

   "http://148.202.89.58/site/?nom=")):
              print('Conectado, subiendo
                     respaldo...')
              mover()
              n= 0
             break
           "BD_resp.py" [Neexcept:
                        print('Aun         sin
   conexion')
            n = 0
    while True:
        try:
            # Retries 'max_tries' from DHT11
    to get a valid result
```

Caja 8. Código de operación en la Raspberri Pi (cont. I)

```
        r  =  dht11.get_result(max_tries=10)    #
'max_tries' defaults to 5
        unix= int(time.time())
        date=
str(datetime.datetime.fromtimestamp(unix).strfti
me('%Y-%m-%d %H:%M:%S'))
        if r:
            print('Temp:   {0:0.1f}   C   Humedad;
{1:0.1f} %'.format(r[0], r[1]))
            i= i+1
            if e==True:
                cur.execute('INSERT          INTO
Site1(Fecha,    Temperatura,    Humedad)    VALUES
(?,?,?)',(date, r[0], r[1]))
                contents                        =
urllib.request.urlopen("http://148.202.89.58/sit
e/?nom="+str(site)+"&temp="+str(r[0])+"&hum="+st
r(r[1])).read()
                e=False
            elif r[0]>=31 :
                cur.execute('INSERT          INTO
Site1(Fecha,    Temperatura,    Humedad)    VALUES
(?,?,?)',(date, r[0], r[1]))
                contents                        =
urllib.request.urlopen("http://148.202.89.58/sit
e/?nom="+str(temp)+"&temp="+str(r[0])+"&hum="+st
r(r[1])).read()
                time.sleep(5)
            elif r[1]>=80 :
                cur.execute('INSERT          INTO
Site1(Fecha,    Temperatura,    Humedad)    VALUES
(?,?,?)',(date, r[0], r[1]))
                contents                        =
urllib.request.urlopen("http://148.202.89.58/sit
e/?nom="+str(hum)+"&temp="+str(r[0])+"&hum="+str
(r[1])).read()
                time.sleep(5)
            elif i== 1500:
                print('30 min')
                i=0
                cur.execute('INSERT          INTO
Site1(Fecha,    Temperatura,    Humedad)    VALUES
(?,?,?)',(date, r[0], r[1]))
                contents                        =
urllib.request.urlopen("http://148.202.89.58/sit
e/?nom="+str(check)+"&temp="+str(r[0])+"&hum="+s
tr(r[1])).read()
else:
            print('Fallo        al        obtener
resultados!')
                time.sleep(1)
                conn.commit()
        except:
            print('Sin conexion')
            no_conex()
            i = 0
    cur.close()
    cur2.close()
```

Caja 9. Código de operación en la Raspberri Pi (cont. II)

Una vez implementado dicho sistema, se procedió a instalarlo en el Data Center, para su funcionamiento y

conexión a un puerto Ethernet y quedando con la dirección ip 148.202.XX.57, como se puede observar en la figura 9.

Figura 9. Data Center con la raspberry y el DHT11

Iniciando la lectura de los datos y su almacenamiento tanto a la base de datos local como a la permanente en la nube. Se puede ver en la figura 10, como se almacenan los datos en la BD de la nube:

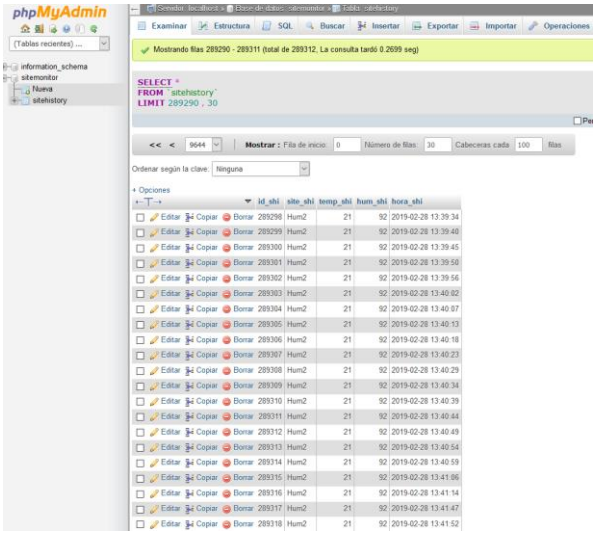

Figura 10. Historial de la DB de la Nube

Una vez funcionando bien la URL y obteniendo la consulta de la DB, como consiguiente se agregó a la app en desarrollo, como se muestra más adelante.

Aplicación Móvil

Desarrollando en el ambiente de programación de Android Studio [16] oficial para la plataforma Android, que se basa en IntelliJ IDEA, figura 11, que básicamente es un entorno de desarrollo integrado para el desarrollo de programas informáticos., Android Studio cuanta con muchas características las cuales permiten agilizar el trabajo como la instalación de los apk de manera instantánea con Instant Run, el cual realiza cambios en la aplicación mientras la app se ejecuta, además cuenta con un entorno unificado en el que se puede desarrollar para todos los dispositivos Android entre otras grandes funciones a resaltar es que cuanta con el soporte incorporado para Google Cloud Platform, lo que facilita la integración de Google Cloud Messaging y App Engine.

Figura 11. Inicio del programa Android Studio [13]

Se creó el proyecto con el software previamente instalado, usando el Appi 19: Android 4.4 (KitKat), ya que es el sistema mínimo con el que la mayoría de los dispositivos cuenta y se pueden desarrollar e implementar diversos elementos que hacen más versátil la app, como animaciones, entre otras.

Se seleccionó la opción Empty Activity como se aprecia en la figura 12, el proyecto vacío, en esta app no se desarrollará con muchos estilos debido que nuestro objetivo es la efectividad con la que esta realice la visualización de los datos.

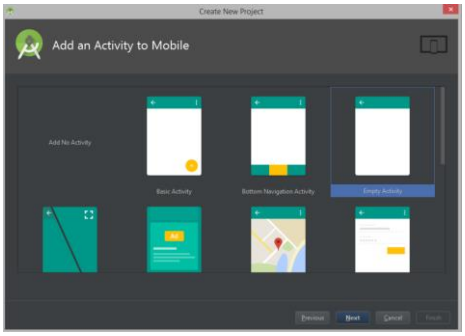

Figura 12. Empty Activity

Por medio de la vista de diseño obtenemos la siguiente vista con cada EditText con un Id, mostrando al usuario los campos que serán llenados por los registros de la DB, tal como se muestra en la figura 13.

Figura 13. Vista de usuario

La figura 14, muestra el método para poder obtener los datos convirtiéndolos a una entrada de caracteres, por el método Get dando un límite de caracteres de la URL y dando una respuesta a la obtención de datos con la conexión HttpURLConnection abriéndolo y limitando rangos de conexión y lectura, después devuelve un código de confirmación, de que se efectuaron con éxito las acciones, almacenándolos en el buffer que es el input String que se creó anteriormente.

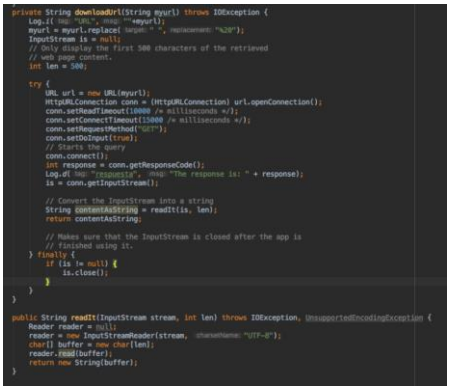

Figura 14. Obtención de URL

Los métodos de los botones se crean en una AsyncTask el cual permite el uso adecuado del subproceso de interfaz de usuario. Para finalizar la figura 15, muestra los campos obtenidos de la DB, con la app previamente instalada en el SmartPhone funcionado sin error alguno.

Figura 15. Datos obtenidos de la DB en la app

IV. Conclusión

En este artículo se presenta la implementación de un sistema de monitoreo de temperatura y humedad basado en una raspberry Pi, usando tecnología IoT, que recolecta y envía información del sensor de manera periódica del DC, a través de una base de datos local, a la nube del CUValles. Asimismo, el sistema incluye herramientas de visualización, monitoreo, y alarma que permiten acceder a la información de manera remota, desde cualquier dispositivo móvil.

Trabajo futuro

Se le añadirán más tareas a la raspberry Pi 3B+, para un mejor monitoreo y control del DC, ya que por el momento está un poco sobrada, ahora se trabaja en la instalación de una chapa electrónica para el acceso controlado al DC, así también se trabaja en la conexión de una cámara, que será activada cuando se abra la puerta de acceso al DC y transmitirá video de lo que esté sucediendo ahí, usando la misma raspberry Pi.

Agradecimientos

Agradecemos al CUValles de la Universidad de Guadalajara por el espacio proporcionado en sus laboratorios, a la Maestría en Ingeniería de Software quien proporcionó recursos a través de su Proinpep y al programa de Verano de la Investigación Científica y Tecnológica del Pacífico, por el apoyo a través de los jóvenes estudiantes que participaron en este programa.

Referencias

[1] Evans Dave. (2011), The Internet of Things, How the Next Evolution of the Internet Is Changing Everything, Cisco.
[2] Shancang Li & Li Da Xu & Shanshan Zhao, The internet of things: a survey, Springer Science+Business Media New York, Published online 26 April 2014.
[3] ETSI. (2013). The European Telecommunications Standards Institute, [cited 2013 May 20]; available from http://www.etsi.org/.
[4] WikiPedia, La enciclopedia libre. Red de Sensores, [On line]. Available from https://es.wikipedia.org/wiki/Red_de_sensores.
[5] Data Centers Hoy, Protección y administración de datos en la empresa [On line]. Available: http://www.datacentershoy.com/2013/07/cual-es-la-temperatura-correcta-de-un.html. [Accessed: 14-dicember-2018].
[6] Raspberry Pi, A small and affordable computer that you can use to learn programming, [On line]. Available: https://www.raspberrypi.org/. [Accessed: 14-dicember-2018].
[7] El Confidencial, Dos millones de razones para saber qué es exactamente Raspberry Pi, [On line]. Available: https://www.elconfidencial.com/tecnologia/2013-11-22/dos-millones-de-razones-para-saber-que-es-exactamente-raspberry-pi_56003/. [Accessed: 14-dicember-2018].
[8] Electronilab, [On line]. Available: https://electronilab.co/tienda/sensor-de-temperatura-y-humedad-dht11/. [Accessed: 14-dicember-2018].
[9] Python, [On line]. Available: http://www.python.org, [Accessed: 14-dicember-2018].
[10] SQLite, [On line]. Available: https://www.ecured.cu/SQLite. [Accessed: 18-dicember-2018]
[11] SQLite: La Base de Datos Embebida, Publicado en SG #17, HERRAMIENTAS Y TECNOLOGÍAS, Autor Filein Rómmel, https://sg.com.mx/revista/17/sqlite-la-base-datos-embebida.
[12] SQLite, [On line]. Available: https://www.sqlite.org/features.html. [Accessed: 14-dicember-2018].
[13] MySQl, Oracle, 2018, https://www.mysql.com/products/enterprise/mysql-datasheet.en.pdf
[14] NOOBS, Offline and network install, [On line]. Available: https://www.raspberrypi.org/downloads/noobs/. [Accessed: 14-dicember-2018].
[15] Experiencing IT, Preparing the SD card, [On line]. Available: https://www.experiencingit.net/raspberry-pi/install-raspbian-lite-raspberry-pi-3/. [Accessed: 14-dicember-2018].
[16] Android, [On line]. Available: https://www.android.com/intl/es-419_mx/. [Accessed: 14-dicember-2018].