Sistema integral para el monitoreo y control de los sonidos en bares y zonas céntricas de la ciudad en tiempo real

Juan Villacreses, Lissette Munizaga, Washington Velásquez V.
Facultad de Ingeniería en Electricidad y Computación, Escuela Superior Politécnica del Litoral, Km. 30.5 Vía Perimetral, Guayaquil, Ecuador
(jmvillac, lmunizag, wavelasq)@espol.edu.ec

Resumen. En el presente articulo se describe un sistema de monitoreo de ruido ambiental que podría contribuir a las futuras tomas de decisiones de los legisladores, permitiendo aportar al desarrollo de la ciudad y al bienestar de los ciudadanos ecuatorianos, desarrollado bajo el uso de hardware y software libre captando los decibelios de ruido en el ambiente; estos luego son procesados y posteriormente son mostrados en un gráfico de tiempo real en la plataforma web SIMORA que ha sido implementada en lenguaje Python, emitiendo notificaciones de correo electrónico y SMS a los administradores de las zonas e infractores causantes del exceso de ruido.

Palabras Clave: Ciudad Inteligente, Ruido, Control Judicial, Medición y Control.

1 Introducción

El ruido es la sensación auditiva inarticulada desagradable en el medio ambiente, causando molestias a los seres humanos. Según la Organización Mundial de la Salud, el nivel sonoro medio no debe exceder de los 45 dBA durante la noche y 55 dBA durante el día [1].

Si hay demasiado ruido la salud se ve afectada, puede producir efectos sobre quienes sufren diariamente con este a su cercanía; eso hace que la calidad de vida se resienta causando enfermedades como estrés, hipertensión, falta de sueño y la más grave la hipoacusia (pérdida total o parcial auditiva), en Ecuador la ley orgánica de prevención y control de la contaminación ambiental establece que los niveles permisibles de ruido apto para las actividades humanas no deberán sobrepasar los 55 decibelios [2].

Guayaquil es una ciudad altamente ruidosa, el problema se encuentra en que no existe sanción alguna para los infractores causante del mismo; ya que dentro del pleno de la Asamblea Nacional no han llegado a ponerse de acuerdo entre los legisladores para emitir las respectivas leyes; debido a que no existe un mecanismo que genere pruebas de una contravención para imponer una multa [3]. El sistema de monitoreo y control de ruido es muy favorable para dirigirlo a las organizaciones encargadas como los Gobiernos Autónomos Descentralizados Municipales, Asamblea y Gobierno Nacional y así colaborar en la continuidad del proceso de poseer ciudades inteligentes precautelando el bienestar y buen vivir de los habitantes, incrementando atractivos turísticos que permitan cooperar en el creciente desarrollo del país [4].

2 Implementación

El principal servicio está enfocado en captar los decibelios ambientales emitidos en las zonas céntricas y bares de Guayaquil, siendo estos posteriormente procesados dentro de una base de datos y pagina web, luego estos al ser sobrepasados de los rangos permitidos por las leyes del Ecuador, son enviados a los administradores de las zonas y a los respectivos causantes del ruido en la ciudad, para que tengan constancia de las faltas cometidas mediante la generación de reportes a correos y celulares personales.

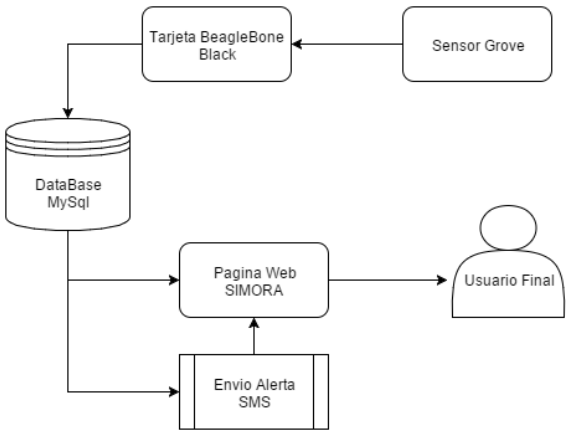

Fig. 1. Diseño Esquemático del desarrollo del Sistema de control y monitoreo de ruido

En el diseño esquemático (Fig. 1) se detalla la ubicación del sensor de ruido que debe ser en lugares estratégicos de la ciudad para que capte los niveles generados en la zona analizada, este sensor debe mantener una conexión a la tarjeta BeagleBone Black [5], esta vendría hacer el controlador de los sensores, la programación y los datos generados en tarjeta son almacenados en la base de datos y luego estos serán presentados en la página web; así como al mismo tiempo son originados los mensajes SMS usando el API de Twilio.[6]

2.1 Manejo electrónico

La tarjeta BeagleBone puede ser alimentada de cualquiera de estas dos formas, la primera mediante su Jack de 5 Voltios de Corriente Directa (DC) cuya capacidad de corriente dependerá la fuente conectada por este medio, y la segunda forma de alimentar la tarjeta es mediante el puerto USB tipo A cuyo límite máximo de entrega de corriente es de 0.5 Amperios (A).

Es importante indicar que no se tiene la obligación de realizar la instalación de Drivers en la BleagleBone Black; ya que viene precargado con el sistema operativo DEBIAN, en el manejo electrónico se realiza las debidas y correctas conexiones del sensor de ruido a los puertos físicos y lógicos de la tarjeta, que son:

- *Salida Analógica*: El pin macho de salida analógico del sensor de ruido puede ir conectado a los pines hembras **33, 35, 36, 37, 38, 39, 40** de la tarjeta BeagleBone Black, que son los designados para los puertos analógicos (AIN – Analog Input).
- *Tierra Física*: El pin macho de Tierra Analógica del sensor, va conectado al pin hembra **34** de BeagleBone designada para el GND (ground).
- *Alimentación Positiva de Tensión*: El pin macho de Voltaje de Corriente Continua (VCC+) del Sensor de Ruido, podría ir conectado a los pines hembra **3, 4, 5, 6** de la tarjeta BeagleBone que corresponden a la tensión positiva en la tarjeta, pero es importante indicar que los pines **3, 4** manejan únicamente un voltaje de 3.3 DVC; mientras que el sensor requiere una alimentación de 5DVC y por este motivo se escoge los pines **5** o **6** que manejan dicho voltaje.

A continuación, en la **Fig. 2** se muestran los pines elegidos en la implementación:

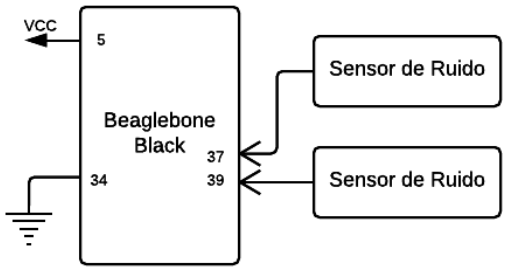

Fig. 2. Conexiones de los sensores de ruido en la Beaglebone Black

2.2 Manejo programable

Se realizó la efectiva programación para controlar los datos obtenidos desde el sensor de ruido; así como también la debida proyección para poder emitir las notificaciones GSM. El código principal del sensor de ruido está encargado de manejar el tiempo a esperar por sonidos en segundos, el control de los voltajes del sensor para enviar la señal receptada a la base.

La base de datos utilizada es MySQL, se ha recopilado la información necesaria realizando la identificación de los objetos u entidades más importantes; una vez establecidos los candidatos para las tablas, se identifica los tipos de información para cada uno de ellos. Se menciona que entre los factores principales para la generación de la misma son: Usuarios, Alerta por defecto, Programar Alarma, Alarma Detectada, Control, Localidad y Supervisor.

Una forma simple de acceder a la base de datos es como se muestra a continuación:

```
import MySQLdb
import datetime
class MyDB(object):
db_connection = None
db_cur = None
def __init__(self):
self._db_connection                                                    =
MySQLdb.connect('192.168.1.137','casa','casa','simora' )
self._db_cur = self._db_connection.cursor()
def set_mV(self, mV, loc_id, fecha):
q  =  "INSERT  INTO  home_sensor(mV,fecha,  localidad_id)  VALUES
(%s,'%s',%s)" % (mV, fecha, loc_id)
self._db_cur.execute(q)
self._db_connection.commit()
def set_alarm(self, start_datetime,end_datetime, loc_id):
q = "INSERT INTO home_alarm (visto, start_datetime, end_datetime,
localidad_id)   VALUES   (0,'%s','%s',%s)"   %   (start_datetime,
end_datetime, loc_id)self._db_cur.execute(q)
```

Entre las funciones más utilizadas de la tarjeta BeagleBone y programadas en el lenguaje de Python se detallan los siguientes:

- **Insertar Datos**

```
import Adafruit_BBIO.ADC as ADC
import MySQLdb
db = MySQLdb.connect("192.168.1.137","root","bbb","tesis_db" )
cursor = db.cursor()
ADC.setup()
analogPin="AIN5"
while(1):
signalMax = 0
signalMin = 1800
for index in range(10):
potVolt=ADC.read_raw(analogPin)
if potVolt > signalMax:
signalMax = potVolt
elif potVolt < signalMin:
signalMin = potVolt
peakToPeak = (signalMax - signalMin)+100;
sql_insert  =  "INSERT  INTO  home_sensor(mV,fecha,  localidad_id)
VALUES (%s,now(),1)" % peakToPeak
try:
cursor.execute(sql_insert)
db.commit()
db.rollback()
```

Este código permite el ingreso de valor del sensor; entre cada (10) muestras, y se obtiene el valor máximo y mínimo de la señal; luego mediante una resta de ambos valores se adquiere la amplitud de la señal recibida.

• Envío de notificaciones mediante Correo Electrónico

```
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
def alert_email():
fromaddr = "juan_men10@hotmail.com"
toaddr = ["juanx90@msn.com", "jmvillac@espol.edu.ec"]
msg = MIMEMultipart()
msg['From'] = fromaddr
msg['To'] = ", ".join(toaddr)
msg['Subject'] = "SUBJECT OF THE MAIL"
body = "YOUR MESSAGE HERE"
msg.attach(MIMEText(body, 'plain'))
server = smtplib.SMTP('smtp-mail.outlook.com', 25)
server.starttls()
server.login(fromaddr, "contrasena")
text = msg.as_string()
server.sendmail(fromaddr, toaddr, text)
server.quit()
```

Código que emite notificaciones de las alertas presentadas al correo del usuario loggeado, y a las demás direcciones de la lista de correos de usuarios infractores.

• Envío de Notificaciones mediante GSM

```
from twilio.rest import TwilioRestClient
# Your Account Sid and Auth Token from twilio.com/user/account
account_sid = "ACd48d7cada5149244f0abda0a4c002e68"
auth_token  = "43a2a3079d2f2d26809ccb0951f4c6c0"
client = TwilioRestClient(account_sid, auth_token)
message = client.messages.create(body = "Ha recibido una alerta
del Sistema de Control SIMORA”,
to="+593968984786",    # número telefónico de administrador SIMORA
from_= "+593986245176") # número telefónico del infractor
print message.sid
```

Este código emite las respectivas notificaciones de alertas vía SMS a los usuarios con la sentencia *client.messages.create*, este método recibe el texto del mensaje, el número emisor y el receptor, mediante una página web de suscripción de mensajería llamada *twilio.com*.

Se cuenta con un sistema de plantillas que procesa los archivos, donde se combina HTML con la lógica de programación en Python haciendo dinámico al sitio web, el uso del framework web Django de Python proporcionó a la implementación de la aplicación una manera más rápida y con menos código. La función *view (vista)* de Python se encarga de tomar una petición web y devuelve una respuesta del mismo tipo, siendo estas: el contenido HTML de la página creada, una redirección URL, error 404, documento XML o Jason Response. En esta función se almacena el contexto (diccionario) de Python que será reenderizado en las plantillas.

Se consideró un servidor de código abierto para enviar páginas web estáticas y dinámicas, que sea multiplataforma, modulares, extensibles y populares, en el cual se pudo implementar el Sistema de Ruido Ambiental que vaya almacenando la información. Se recalca que para poder realizar la interconexión entre el servidor y la aplicación web se hace uso de una interfaz simple en donde se utilizó WSGI [7].

3 Resultados

Se logró implementar un sistema que permite mejorar el buen vivir de los habitantes cercanos a zonas de bares y traficadas que bajo investigaciones previamente realizadas han debido soportar durante mucho tiempo el malestar del exceso de ruido; el sistema permite sensar los niveles dentro de Guayaquil ocasionados por los diversos factores**,** con algunos funcionamientos que ayudaran a los usuarios tanto administradores, como autoridades pertinentes.

En las capturas de pantalla se detalla los resultados obtenidos dentro del proceso realizado en la zona de análisis. Se muestra dentro de la página web el gráfico de decibelios en tiempo real, donde se logra observar que los "picos" de la gráfica es el exceso de rango en las mediciones tomadas estando estas previamente establecidas y configuradas en el sistema por niveles y horarios. Este gráfico fue captado dentro de un rango de prueba de 3 minutos en el sector, donde se muestra 3 excesos de decibelios *(picos mayormente elevados).*

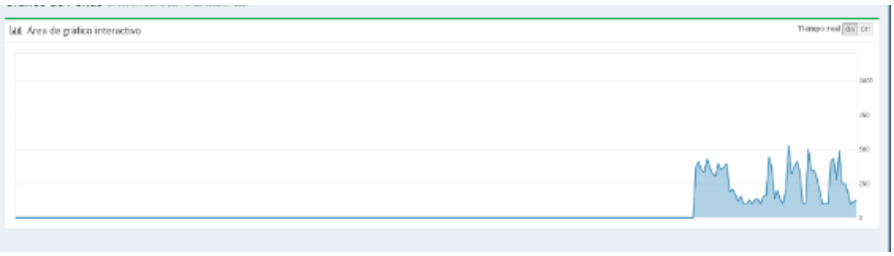

Fig. 3. Gráfico de alarma generada en tiempo real al sobrepasar los límites de decibelios configurados en el Sistema

Cabe decir, que los picos que maneja el sistema están definidos para captar todo el ruido posible en las zonas que se desean controlar, por ejemplo: Si el ruido se produce por 1 segundo, el sistema guarda el evento pero no alerta al administrador, lo contrario, si el ruido es producido por un tiempo considerable (configurable en el sistema) se alerta al encargado para que tome las medidas necesarias. La aplicación también emite un gráfico seccionado en rangos de niveles, donde se presenta el porcentaje de ruido que generó una alarma en varios intervalos de tiempo. Se observa en la Fig. 4. los

porcentajes de rangos generados por la alarma y en un 34% el máximo fue perteneciente a 500 – 963 mili voltios

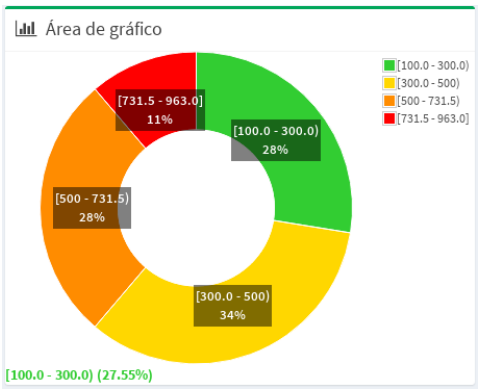

Fig. 4. Porcentaje máximo de mili voltios generados por una alarma.

Se desea llevar un registro de todas las alarmas generadas por el infractor; de esta manera al ser reincidente se tienen las evidencias del caso. En la Fig. 5. se puede visualizar las alarmas generadas, la toma de muestras presenta un frecuencia de 4 cada de 10sg (enviando el promedio al servidor); ya que se pueden generar altibajos de ruido dentro de un tiempo menor al mencionado y estos pasan a ser considerados en el muestreo.

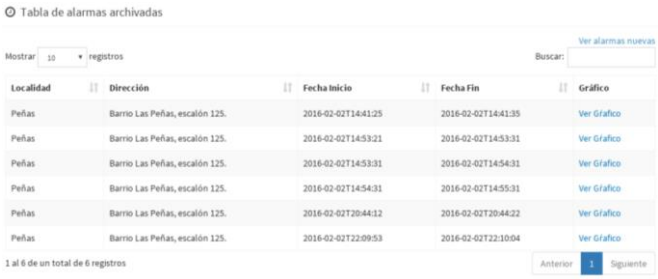

Fig. 5. Alarmas almacenadas por localidad, fechas y horarios que fueron generadas en la Zona de prueba.

Las notificaciones emitidas a los administradores e infractores se hacen mediante dos medios siendo estos correos electrónicos y SMS a los celulares personales de los usuarios registrados, como se lo demuestra en los resultados detalladamente en la Fig. 6., se mantiene una emisión de mensajes GSM permaneciendo suscrito a twilio, y el administrador recibe una alarma generada del sector analizado. Así mismo, el sistema permite generar un reporte de todas las infracciones, para que conste como evidencia de las autoridades.

Fig. 6. Notificación GSM recibida por alarma generada

3.1 Análisis de comunicación

Para comprobar la factibilidad del sistema se plantaron dos nodos sensores en la ciudad de Guayaquil, uno en el barrio las "Peñas" y otro en la calle 9 de octubre y Esmeraldas. La información obtenida se presenta a continuación:

Nodo	**Tiempo de envío de la notificación**	
	Email	**SMS**
Peñas	10 s	15 s
9 de Octubre y Esmeraldas	15 s	16 s

Tabla 1. Tiempo de envío de la notificación a los administradores desde diferentes zonas de la ciudad

En la **Tabla 1**, se presenta el tiempo promedio en que las notificaciones son recibidas por los administradores, como se puede observar el *email* llegará primero en el caso de una incidencia con un promedio de 10s por correo, el tiempo de retardo que existe en el envío de los *SMS* se debe a un delay presente en el código y a su vez, la invocación del API de Twilio quien transmite el mensaje.

Nodo	**Número de paquetes enviados a la aplicación**			
	Recibidos / hora		**Perdidos / hora**	
	Día	**Noche**	**Día**	**Noche**
Peñas	13202	10045	1403	3125
9 de Octubre y Esmeraldas	8402	13023	4300	1203

Tabla 2. Paquetes enviados en el Día (9:00 am) y en la Noche (9:00 pm) a la aplicación en diferentes zonas de la ciudad

En la **Tabla 2**, se presenta una comparación entre los paquetes recibidos y perdidos en función del tiempo, para esta prueba los nodos estuvieron encendidos "*1 hora*" por cronómetro, se aprecia que existe una variación considerable cuando se realizan las pruebas en el día como en la noche y a su vez en la ubicación donde se encuentra el nodo. Por ejemplo: cuando se transmite en las "*Peñas"* por la mañana el flujo de paquetes es más limpio, lo contrario ocurre en la noche donde existe una variación debido al ruido ambiental. Por otro lado, en "*9 de Octubre y Esmeraldas*" se aprecia lo opuesto a las Peñas, esto debido a que en las mañanas el tráfico vehicular es más afluente en relación a las noches.

Tabla 3. Paquetes enviados en ambiente de laboratorio a la aplicación (1 hora)

Nodo	**Número de paquetes enviados**	
	Recibidos / hora	**Perdidos / hora**
Laboratorio	14389	10

En la **Tabla 3**, se presenta una prueba en ambiente de laboratorio que permite realizar una comparación entre un ambiente real y uno simulado, como se puede apreciar los escenarios que más se asemejan son las *Peñas* en el día y 9 *de Octubre y Esmeraldas* en la noche. Así mismo, lo que se pudo observar al comprobar los gráficos del ruido ambiental en cada zona es que no se veían alterados por la pérdida de paquetes como para perjudicar el rendimiento del sistema.

4 Conclusiones

El sistema propuesto presenta una solución viable para el control de ruidos en las principales zonas de Guayaquil, aportando una mejora en la calidad de vida de los habitantes, dado que se detectó cuáles son las necesidades reales de las personas que habitan y militan con los molestos problemas; los procesos operativos del sistema procreado se apegan a la realidad de la convivencia, proveyendo beneficios de salud y del buen vivir.

Se plantea a futuro integrarlo con el sistema de seguridad de cámaras de vigilancia ECU-911 que actualmente es utilizado por las organizaciones de emergencia como Policía Nacional, Cuerpo de Bomberos, para proveer un mejor resguardo a la ciudadanía Guayaquileña y del Ecuador. Se estima realizar estudios de localización previa a la colocación e instalación de los sensores para prever que no existan daños de los mismos a causa de los factores ambientales como sol y lluvia.

Considerando la reducción de costos y tomando en cuenta que la ciudad cuenta con un proyecto de "Digital City" con múltiples puntos de redes de Internet ya instalados; se estudia la utilización de dichas redes inalámbricas para el uso de la tarjeta programable y realizar la obtención de los resultados deseados.

Referencias

1. Diba, Organización Mundial de la Salud, [En línea]. Available: https://www.diba.cat/c/document_library/get_file?uuid=72b1d2fd-c5e5-4751-b071-8822dfdfdded&groupId=7294824. [Último acceso: 25 Octubre 2015].
2. Asamblea Nacional, Ley de control de la contaminación ambiental, [En línea]. Available: http://ppless.asambleanacional.gob.ec/alfresco/d/d/workspace/SpacesStore/18723cc9-1710-41d8-ac3c-036f4be09989/Proyecto%20de%20Ley%20Reformatoria%20a%20la%20Ley%20de%20Prevenci%C3%B3n%20y%20Control%20de%20la%20Contaminaci%C3%B3n%20Ambiental%20Tr.%2050670.p. [Último acceso: 25 Octubre 2015].
3. Debian, Sistema Operativo Debian, [En línea]. Available: https://www.debian.org/releases/jessie/mips/ch01s01.html.es. [Último acceso: 7 Diciembre 2015].
4. BeagleBoard, Puertos de conexión BeagleBone, [En línea]. Available: http://beagleboard.org/support/bone101. [Último acceso: 8 Enero 2016].
5. Diba, Sensor de ruido Grove, [En línea]. Available: https://www.diba.cat/c/document_library/get_file?uuid=72b1d2fd-c5e5-4751-b071-8822dfdfdded&groupId=7294824. [Último acceso: 14 Octubre 2016].
6. API de Twilio, SMS APIs for Text Messaging, VoIP & Voice in the Cloud – TWILIO, [En linea], Available: https://www.twilio.com/
7. WSGI, Programación WSGI_Django, [En línea]. Available: http://wsgi.tutorial.codepoint.net/. [Último acceso: 14 Noviembre 2015].