Domótica básica en un hogar a escala utilizando el servidor RabbitMQ

Emmanuel De Jesús Velásquez Martínez
Universidad Autónoma de Zacatecas
Unidad Académica de Ingeniería Eléctrica.
Carretera Zacatecas – Guadalajara Km. 6, Ejido la Escondida, C.P. 98160
zefeca27@gmail.com

Dr. Roberto Solís Robles
Universidad Autónoma de Zacatecas
Unidad Académica de Ingeniería Eléctrica.
Carretera Zacatecas – Guadalajara Km. 6, Ejido la Escondida, C.P. 98160
rsolis@uaz.edu.mx

Resumen — En la actualidad es importante computarizar de forma segura procesos comunes como lo es una vivienda asimismo evitar que esta sufra daños por accidentes comunes que se pudieran, es por ello que en este proyecto se plantea desarrollar un prototipo a escala de una vivienda para monitorear y controlar funciones básicas tales como control de iluminación, la temperatura ambiente, seguridad ante intrusos, operación de abrir o cerrar puerta y la calidad del aire es decir control de gases tóxicos esto a través del servidor de mensajería middleware RabbitMQ, cada operación estará en función bajo el uso de sensores, controladores y actuadores previamente desarrollando el circuito propuesto montado en la tarjeta Raspberry PI de modo que estará recibiendo y enviando datos dependiendo de lo que el usuario requiera, asimismo las funciones estarán desarrolladas bajo lenguaje de programación Python, para esta sección del proyecto fungirá como servidor RPC (Llamada a Procedimiento Remoto). Para monitorear y/o accionar los dispositivos electrónicos se realizó una interfaz cliente RPC (Llamada a Procedimiento Remoto) bajo lenguaje de programación Java para que el usuario pueda realizar la interacción con cada una de las funciones propuestas de forma amigable a través de los componentes de la interfaz con su hogar. Cabe señalar que la comunicación entre cada sección (cliente-servidor) está en forma segura mediante el protocolo SSL/TLS con la finalidad de tener privacidad e integridad en la comunicación entre dos puntos en la red de comunicación.

Palabras clave — RabbitMQ, Raspberry PI, interfaz Java, Python, protocolo SSL/TLS, RPC.

I. INTRODUCCIÓN

La palabra domótica proviene de la unión de las palabras *Domus* que significa casa y *Autónomo* que significa que funciona por sí solo. La domótica consiste en dotar de automatización a una casa o edificio para que esta se ocupe por si misma de ciertas tareas, estas tareas pueden ser tan simples como acondicionar la temperatura en una habitación o tan complejas como encargarse de la seguridad de todo un edificio [1].

La domótica se inició a comienzo de la década de los 70, cuando aparecieron las primeras pruebas en pisos piloto. Ya en los 80 fue cuando se empezaron a comercializar los circuitos integrados, fue cuando la domótica se empezó a expandir al hogar. Posteriormente la domótica consigue integrar dos sistemas (el eléctrico y el electrónico) y en enseguida la comunicación integral entre los dispositivos del hogar [2].

Las principales funciones de la domótica de un hogar son:

- Control local y remoto de la iluminación en la vivienda.
- Iluminación por detección de presencia.

- Automatización de persianas y toldos.
- Control y gestión de energía.
- Acceso electrónico al hogar (porteros digitales).
- Control y monitoreo de alarmas técnicas como detección de fugas de agua, gas, humo.
- Sistema de mensajería si algo sucede en el hogar.
- Realización de acciones preventivas automáticas: cierre de persianas, corte de la energía, entre otros.
- Climatización.
- Control del aire acondicionado para regular la temperatura dentro de la vivienda.
- Control de riego.
- Control y diagnóstico de electrodomésticos y ahorro de energía.
- Encendido y apagado remoto de electrodomésticos.

Para llevar a cabo las tareas mencionadas con anterioridad el sistema tiene que estar formado principalmente por tres dispositivos sensores, controladores y actuadores:

- Sensores: son los dispositivos encargados de medir aquello que se quiere controlar.
- Controlador: es quien decide que se debe hacer en función de lo que mide el sensor y ciertas reglas predefinidas.
- Actuadores: son los encargados de modificar lo que se está controlando [3].

El sistema básico domótico desarrollado es capaz de recoger información proveniente de sensores, procesarla y emitir órdenes según se vayan solicitando. El sistema está limitado a acceder a redes exteriores de comunicación o información para este contexto se desarrolló bajo una red LAN (Red de Área Local) haciendo uso del servidor RabbitMQ donde este es un estilo de comunicación de servicio a servicio, permite que las aplicaciones se comuniquen enviándose mensajes entre sí. La arquitectura básica de una cola de mensajes es simple, se dispondrá de una interfaz cliente que crea mensajes para la interacción de los componentes electrónicos y los entrega a la cola de mensajes adicionalmente se tendrá otra aplicación de servicio, para el prototipo propuesto se estará ejecutando dicho servicio en una Raspberry PI el cual recuperara el mensaje de la cola y procesara las solicitudes y la información contenida en el mensaje como resultado realizar una acción con cada uno de los componentes electrónicos que contiene la vivienda.

El valor añadido para la domótica es tener el control y la dirección desde una única interfaz, esta interfaz de usuario se usará por medio de una PC el cual señalizara situaciones de peligro y se monitorea los datos y estados esenciales de los componentes electrónicos del sistema diariamente, estas situaciones se limitan a la detección de gases y detección de intrusos. Conjuntamente permitirá la monitorización, señalización y estados

de temperatura, luminosidad de los focos led y abrir y cerrar una puerta, esto para mejorar el confort de las personas que habitan.

Como proyecciones a futuro se espera que este tipo de sistemas cuenten además con algún método que permita la modificación de las reglas definidas en el controlador para permitir adaptar y añadir de forma sencilla las funciones de la casa a nuestras necesidades o gustos en forma local o remota. Este control se puede hacer desde dispositivos específicos como un ordenador o incluso desde un smartphone.

II. MÉTODO

Diseño de la vivienda domótica

La figura 1 muestra la interacción de cada una de las secciones que conforman el proyecto de control domótico, la programación se desarrolló en Java (cliente) y Python (servidor) de tal manera que el servidor RabbitMQ tenga una queue RPC [4] de mensajes. La interacción entre cada módulo inicia cuando el usuario mediante la interfaz gráfica solicita en la vivienda realice una acción con un componente electrónico (1) posteriormente el mensaje enviado (2) y enseguida es colocado en una queue RPC especifica con id de correlación en el servidor RabbitMQ (3), consecutivamente el servidor RPC recibe dicho mensaje con el id de correlación (4) para así realizar una acción en los dispositivos de la vivienda (5), una vez que se realice dicha acción este responde al cliente con el id de correlación con el que fue enviado el mensaje (6) para que de esta forma en la interfaz dependiendo de la acción realizada pueda procesar la información y mostrarla en la interfaz (7) dependiendo del estado de dispositivo (por ejemplo si encendió un foco led el cliente debe recibir del servidor que efectivamente el foco led esta encendido). La figura 1 mostrada representa una red LAN, entonces para establecer comunicación para envíos y recepciones de mensajes entre ambos extremos a cada sección se le asigno una dirección IPv4, la parte de interfaz de cliente RPC y del servidor RabbitMQ se le asigno la dirección 192.168.1.10/24 y para la parte del servidor RPC la dirección 192.168.1.20/24 ambas direcciones pertenecientes a una misma red.

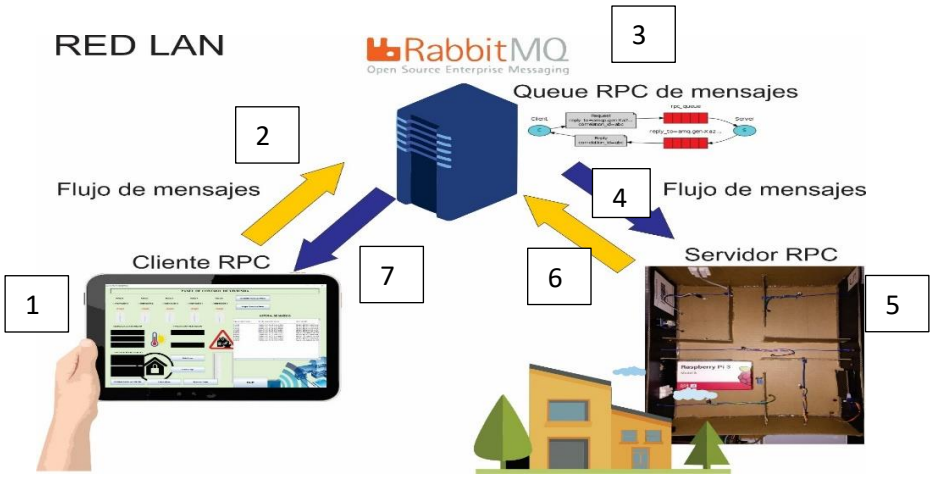

Figura 1. Diagrama de interacción entre cliente/servidor RPC a través del servidor RabbitMQ

Servidor RPC (Llamada a Procedimiento Remoto) – Maqueta simulando vivienda real domótica

Para la parte del servidor RPC se realizaron las conexiones de todos los componentes electrónicos con los que se interactuaran para realizar una función específica cada uno en su categoría. La tarjeta Raspberry PI cuenta con GPIO's lo que significa Entrada/Salida de Propósito General, como se muestra en la figura 2 la enumeración de cada GPIO dependerá del modelo de la tarjeta Raspberry PI en este caso corresponde a los GPIO´s de una Raspberry PI 3 B+. Los GPIO´s realizan una función dependiendo de la configuración establecida en el lenguaje de programación en este caso Python, estos pueden recompilar datos a través de la detección de eventos generados por los sensores o bien cambiar el estado de algún componente electrónico [5].

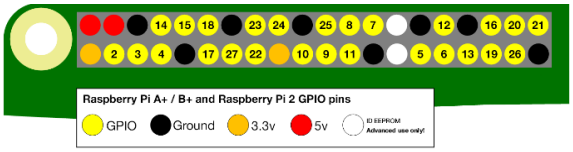

Figura 2. GPIO´s de Rasberry

A continuación, se muestran en la figura 3 el circuito de conexiones de los componentes a través de los GPIO el cual mediante el envío y recepción de los mensajes generados por la interfaz Java y mediante la queue de mensajes del servidor RabbitMQ estarán interactuando cada uno de los dispositivos.

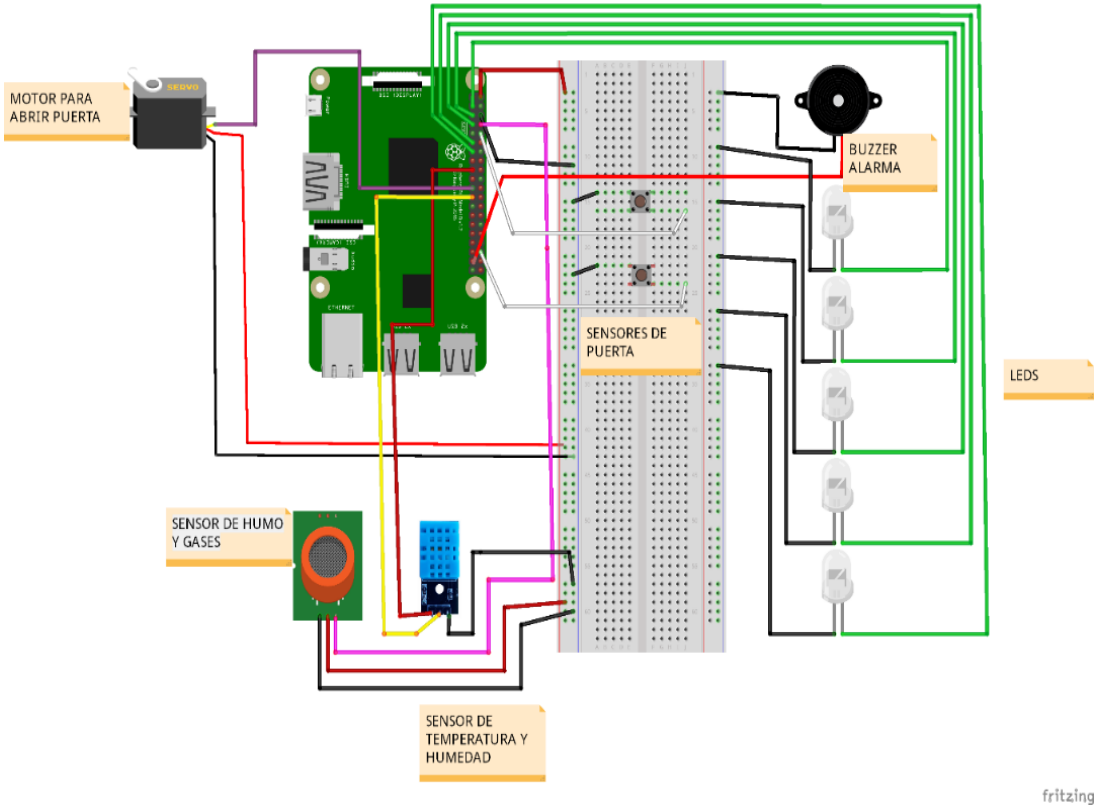

Figura 3. Conexión de componentes electrónicos

Cada componente requiere de un nivel de voltaje para realizar su función, además estos son clasificados de entrada y salida tal y como se listan en la tabla 1:

Tabla 1. Especificaciones de componentes electrónicos

Entrada	Voltaje	Salida	Voltaje
Sensor de gas MQ2	5 V	Focos led	3.5 V
Sensor de puerta	5 V	Buzzer	3.5 v
Sensor de temperatura	3.5 V	Servo motor	5 V

En la figura 4 se muestran los módulos que se utilizaron para realizar funciones:

- pika: su función es establecer una conexión con el servidor RabbitMQ y así poder generar la queue para el envío/recepción de mensajes.
- sys: se utilizó este módulo para poder detectar eventos de señales que son generadas durante la ejecución del programa.
- time: módulo utilizado para hacer uso de la función sleep para que el elemento buzzer emita o apague el sonido en un lapso de tiempo dado en unidades de segundo.
- Adafruit_DHT: el uso de este módulo es poder censar y arrogar la temperatura y humedad captada a través del sensor DHT11.
- threading: el uso de las funciones de esta librería es para crear procesos de forma simultánea, en este caso se requiere que se este ejecutando un proceso capaz de censar y enviar los datos a la interfaz de cliente cada determinada hora, de igual forma con el sensor de gas y con la alarma de detección de intrusos cuando este activa.
- ssl: se implementó este modulo con el fin de crear una conexión segura entre el cliente y el servidor.
- RPi.GPIO: es un módulo para la configuración de los GPIO's de la tarjeta Raspberry PI.

```
import pika, sys, time, Adafruit_DHT, threading, ssl
import RPi.GPIO as GPIO
```

Figura 4. Librerías utilizadas para establecer funciones

Para el manejo de los elementos electrónicos de entrada y salida mediante los GPIO fue necesario escribir en el programa las líneas de instrucciones que se muestran en la figura

5, la primera línea permite establecer que si el canal está en uso este se configure independientemente de la función a realizar para deshabilitar las advertencias. La segunda línea permite el modo de configuración de los GPIO's de la tarjeta Raspberry PI el cual pueden ser de dos formas:

- BOARD: especifica que se está refiriendo a los pines por su número, es decir los números impresos en nuestra Raspberry PI.
- BCM: se refiere a los pines por su número de "Broadcom SOC channel", estos no son correlativos como en el modo BOARD [8].

```
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
```

Figura 5. Configuraciones iniciales para uso de GPIO´s

Los GPIO's se pueden configurar de entrada o salida dependiendo del elemento electrónico que este configurando en lenguaje de programación Python:

Por ejemplo en la figura 6 las primeras dos líneas de instrucciones es un ejemplo para establecer que los dispositivos electrónicos como lo son los leds, el servo motor y buzzer se configuren los GPIO como salida esto pasando como parámetros el número del pin de la tarjeta Raspberry PI y GPIO.OUT (que significa en modo salida), la tercera línea de código es un ejemplo para establecer que se estarán recibiendo señales de los sensores a través del número de pin establecido de la tarjeta Raspberry PI, y posteriormente las siguientes dos líneas de código se usaron para que los sensores magnéticos que se colocaron en los accesos de la vivienda a través de los GPIO´s pueda detectar una fuerza fuerte que en este caso es una imán que baje el voltaje a 0V, en un principio los sensores se habían establecido como GPIO.IN como resultado de esta configuración se generaban falsos pulsos al recibir la señal del sensor magnético lo que provocaba que se mandara un mensaje de intruso falso cuando en verdad no había ocurrido dicho evento, para el manejo de luminosidad de los focos led y el giro del servo motor es necesario establecer un parámetro adicional y es establecer GPIO.PWM el cual es la modulación por ancho de pulsos de una señal es una técnica en la que se modifica el ciclo de trabajo de una señal periódica para controlar la cantidad de energía que se envía [9].

```
GPIO.setup(3, GPIO.OUT)
GPIO.setup(5, GPIO.OUT)


GPIO.setup(8, GPIO.IN)


GPIO.setup(10, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(36, GPIO.IN, pull_up_down=GPIO.PUD_UP)

foco1 = GPIO.PWM(3, 100)
```

Figura 6. Configuración de entrada/salida de GPIO

Para crear la queue de mensajes de forma segura a través del servidor RabbitMQ se escribió en el programa el código de la figura 7, para establecer una comunicación segura entre el envío y recepción de mensajes es necesario especificar los certificados para que se autentique que el cliente al que se responderá los mensajes es efectivamente el cliente, posteriormente para el transporte de mensajes se pasa una lista de parámetros como IP donde esta el servidor RabbitMQ, el puerto (5671 modo seguro), el virtual host, y las credenciales (usuario y contraseña) para esto se creó un usuario nuevo desde la interfaz web de RabbitMQ al que se asoció un virtual host [7,10,14].

```
def proceso(usr, clave, ip, puerto, vh, q):
    contexto = ssl.create_default_context(cafile="/home/pi/Desktop/Proyecto/certsProyecto/testca/ca_certificate.pem")
    contexto.load_cert_chain("/home/pi/Desktop/Proyecto/certsProyecto/client/client_certificate.pem","/home/pi/Desktop/Proyecto/certsProyecto/client/private_k
    ssl_opciones=pika.SSLOptions(contexto, "raspberrypi")

    credenciales = pika.PlainCredentials(usr,clave)
    parametros = pika.ConnectionParameters(ip,puerto,vh,credenciales, ssl_options = ssl_opciones)
    connection = pika.BlockingConnection(parametros)
    channel = connection.channel()
    channel.queue_declare(queue=q)
    channel.basic_qos(prefetch_count=1)
    channel.basic_consume(queue=q, on_message_callback=repuestas)
    print(" [x] Esperando repuestas de '"+q+"'")
    channel.start_consuming()
```

Figura 7. Configuración inicial para el manejo de queue de mensajes

Función para llamar a otra función para la interacción de componentes de la vivienda: la sección de código visualizado en la figura 8 establece mediante la recepción mensaje que se almaceno en su respectiva queue una función a realizar, el mensaje que se envía desde la interfaz de usuario tiene varios parámetros es cual para su separación se usó el carácter '-' donde el primer argumento será la función a ejecutar para la interacción de los componentes y los demás argumentos corresponderán a la tarea específica a realizar al cual responderá mediante el id de correlación con el que se fue solicitado por cada trabajo asociado existe una función [4]. Por ejemplo, en la figura 8 si se quiere interactuar con el control de focos, el primer parámetro debe ser '1' y el segundo pertenece al trabajo a realizar con los elementos electrónicos.

```
def repuestas(ch, method, props, body):
    n = str(body)
    x = n.split('-')
    if(x[0]=='1'):
        n = str(x[1])
        response = encendidoFocos(n)

    ch.basic_publish(exchange='',
                     routing_key=props.reply_to,
                     properties=pika.BasicProperties(correlation_id = \
                                                     props.correlation_id),
                     body=str(response))
    ch.basic_ack(delivery_tag=method.delivery_tag)
```

Figura 8. Segmento de código llamada a la función para interacción entre los componentes

Función para regular luminosidad de focos led: la sección de código mostrado en la figura 9 son instrucciones para el manejo de resplandor del foco led numero 1, la función recibe dos parámetros respecto al mensaje de la queue donde el primer parámetro especifica el foco led de una habitación en específico y el segundo para establecer la luminosidad del mismo mediante la instrucción foco1.start(r) (r se asume un rango 0% como intensidad mínima a 100% como intensidad máxima) una vez que el foco led este regulado según la especificación del usuario este retorna un mensaje con la regulación con la que cuenta el foco para posteriormente mostrar el porcentaje de luz en la interfaz del cliente.

```
def regulacion(n, r):
    if n == "FOCO1":
        foco1.start(r)
        return "FOCO 1 REGULADO AL"+str(r)
```

Figura 9. Código para control de luminosidad

Función de escalas de temperatura y humedad en vivienda: la sección de código mostrado en la figura 10 realiza la conversión en diferentes escalas de temperatura según se solicite, dependiendo de la escala de temperatura se realiza una conversión ya que el módulo Adafruit_DHT arroga por default la temperatura en grados Celsius, por tanto si se requiere la temperatura en otra escala es necesario realizar una conversión de la escala de temperatura solicitada donde finalmente se retorna la fecha y hora conjuntamente de la escala de temperatura y humedad cuyos datos serán retornados y posteriormente visualizados en la interfaz de usuario.

```
def temperatura(n, tipoT):
    now = datetime.now()
    humedad, temperatura = Adafruit_DHT.read_retry(sensor, 23)
    if n == "TEMPERATURA" and tipoT=='Celsius':
        return str(datetime.now())+"+"+str('Temperatura:{0:0.1f}°+Humedad:{1:0.1f}%'.format(temperatura, humedad))
    if n == "TEMPERATURA" and tipoT=='Fahrenheit':
        temp = (temperatura*1.8)+32
        return str(datetime.now())+"+"+str('Temperatura:{0:0.1f}°+Humedad:{1:0.1f}%'.format(temp, humedad))
    if n == "TEMPERATURA" and tipoT=='Kelvin':
        temp = temperatura + 273.15
        return str(datetime.now())+"+"+str('Temperatura:{0:0.1f}°+Humedad:{1:0.1f}%'.format(temp, humedad))
    if n == "TEMPERATURA" and tipoT=='Rankine':
        temp = ((9*temperatura)/5) + 491.67
        return str(datetime.now())+"+"+str('Temperatura:{0:0.1f}°+Humedad:{1:0.1f}%'.format(temp, humedad))
```

Fórmulas de conversión de temperaturas [11].

Figura 10. Código para obtener la temperatura en diferentes escalas

Función de abrir y cerrar garaje mediante servomotor: el segmento de código mostrado en la figura 11 corresponde para accionar la puerta en modo abierta, mediante la interfaz

gráfica y la intervención del servidor RabbitMQ al recibir el mensaje ‘ABRIR GARAGE’ en automático se ejecuta la instrucción p.ChangeDutyCycle(10.5) el cual acciona mediante un giro en el eje del servo motor, de tal manera que en la maqueta simula que la puerta se ha abierto, para el cerrado de la puerta el valor de la instrucción p.ChangeDutyCycle(n) cambia el valor simulando que la puerta está cerrada.

```
def garage(n):
    if n == "ABRIR GARAGE":
        p.ChangeDutyCycle(10.5)
        return "GARAGE ABIERTO"
```

Figura 11. Código abrir puerta

Función detección de gases tóxicos: esta función es llamada de forma constante para garantizar la seguridad de los habitantes del hogar por lo tanto mediante un hilo (proceso) es ejecutado desde la interfaz de usuario y su respectiva queue de mensajes creada en el servidor RabbitMQ en caso de detectar un evento de gases se notificara el suceso de forma instantánea además de activarse el buzzer para mayor atención de las personas, el código mostrado en la figura 12 hace énfasis en la detección de gases tóxicos a través del sensor de gas mq-2 (véase apéndice) cuyos datos estarán censados en el pin 8 cuando se detecte un gas en automático se retorna el mensaje de advertencia.

```
def humo(n):
    if n == "HUMO":
        if GPIO.input(8):
            return str(datetime.now())+"+SE HA DETECTADO GASES+ACTIVAR"
        return ""
```

Figura 12. Código para detección de gases tóxicos

Función detección de intrusos: para detectar a través de los sensores magnéticos el acceso de intrusos los GPIO’s (10 y 36) asociados se preguntaran si están abierto o cerrado el acceso, para activar la alarma ambos sensores deben estar enviando una señal alta ya que es requisito que los accesos se encuentren cerrados, cuando se activa la alarma desde la interfaz grafica se ejecutara un proceso de tal manera que en el momento en que se detecte en la vivienda un acceso abierto esta retorne al cliente un mensaje de notificación que se ingresó a través de la ventana o puerta según sea el caso junto a ello la hora en que se realizó el acto conjuntamente el buzzer emitirá un sonido dentro de la vivienda para ahuyentar al supuesto ladrón.

```
def alarmaLadron(n):
    global gneral
    flag1 = False
    flag2 = False
    if n == "ACTIVA ALARMA":
        if gneral == True:
            gneral = True
        if(GPIO.input(10)==1):
            flag1 = True
        if(GPIO.input(36)==1):
            flag2 = True
        if flag1 == True or flag2 == True:
            for i in range (1, 5):
                GPIO.output(37, True)
                time.sleep(0.5)
                GPIO.output(37,False)
                time.sleep(0.5)
            gneral = True
            if(flag1 == True):
                return "Puerta+"+str(datetime.now())+"+ALARMA ACTIVADA"
            if(flag2 == True):
                return "Ventana+"+str(datetime.now())+"+ALARMA ACTIVADA"
        else:
            return "ALARMA NO ACTIVADA"
```

Figura 13. Código para detección de intrusos

Función de activación de alarma en caso de gases tóxicos o intrusos dentro de la vivienda: en la figura 14 se muestra el fragmento de código para encender la alarma al detectar accidentes preestablecidos para que los habitantes de la vivienda y de alrededores estén al pendiente y así se pueda atendar cualquier situación de riesgo de forma rápida. El buzzer se activa y desactiva cada medio segundo estas instrucciones se ejecutarán de forma infinita hasta que no se envié el mensaje de desactivación desde la interfaz para salir del ciclo infinito.

```
def alarmaGeneral(n):
    while True:
        while gneral:
            if(gneral==False):
                break
            else:
                GPIO.output(37, True)
                time.sleep(0.5)
                GPIO.output(37,False)
                time.sleep(0.5)
```

Figura 14. Código para activación de alarma

Cliente RPC (Llamada a Procedimiento Remoto) – Lógica de interfaz de la vivienda real domótica

Cada elemento (JCheckBox, JSlider, JButton, JComboBox) de la interfaz genera eventos que envía un mensaje con los parámetros necesarios a una queue específica del servidor RabbitMQ para que en la parte del servidor RPC ejecute la instrucción en el dispositivo y este retorne la respuesta de operación.

En la figura se muestra un segmento de código el cual mediante la selección del elemento JCheckBox envía a una queue el mensaje con la instrucción que va realizar el servidor, una vez que se ejecute la acción en el servidor RPC se habilitará JSlider donde tendrá la función de regular la intensidad del foco que se encendió.

```
if(cb.getText()=="HABITACI\u00d3N 1")
{
        PeticionesCliente("1-ENCENDER FOCO HABITACI\u00d3N 1", "focos");
        slider1.setEnabled(true);
}
```

Figura 15. Evento de jcheckbox

En la figura 16 se muestra la creación del objeto cliente para enviar y recibir mensajes de las queue´s, a partir de la respuesta de retorno desde el servidor RPC este realiza cambio en la interfaz, como ejemplo la etiqueta del foco 1 de la interfaz cambia de color indicando que se llevó a cabo la operación de encendido.

```
try(ClienteJAVA funcion = new ClienteJAVA()) {
    System.out.println(" [x] Solicitando que(" + Mensaje + ")");
    String response = funcion.call(Mensaje, NombreQueue);
    System.out.println(" [.] Que sucedio: '" + response + "'");
    String[] datos = response.split("\\+");
    if(response.equals("FOCO 1 ENCENDIDO"))
    {
        StatusF1.setBackground(Color.GREEN);
        StatusF1.setForeground(Color.GREEN);
        StatusF1.setText("ENCENDIDO");
    }
```

Figura 16. Respuesta retornada desde del servidor RPC

En la figura 17 se escribió el código para especificar un listado de las diferentes conversiones de temperatura a través de un JComboBox, por default la temperatura es solicitada en grados Celsius, entonces en base a un evento de selección del ítem del JComboBox se solicita el tipo de temperatura requerido.

```
estados.addItem("Celsius");
estados.addItem("Fahrenheit");
estados.addItem("Kelvin");
estados.addItem("Rankine");
elHilo1.TipoDeTemperatura(estados.getSelectedItem().toString());
estados.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        elHilo1.TipoDeTemperatura(estados.getSelectedItem().toString());
    }
});
```

Figura 17. ComboBox para seleccionar el tipo de escala de temperatura

En la figura 18 se muestra el código para el proceso que se estará ejecutando cuando la interfaz de usuario inicie, este básicamente estará solicitando a través del servidor RabbitMQ la temperatura de vivienda, cuando el sensor responda a la solicitud mediante el mensaje existe dos métodos de esta clase que servirán para poder retornar el dato y este sea visualizado en la interfaz de usuario. Cabe señalar que este código es muy similar para la detección de gases tóxicos y para la alarma de intrusos solo que el proceso para este inicia cuando el usuario solicite su activación.

```
public void run()
{
   while(true)
   {
      try(ClienteJAVA funcion = new ClienteJAVA()) {
         System.out.println("[x] Solicitando que(TEMPERATURA)");
         response = funcion.call("4-TEMPERATURA-"+tipoTemperatura, "temperatura");
         String[] datos = response.split("\\+");
         if(datos.length==3)
         {
            responseHora = datos[0];
            responseTemperatura = datos[1];
            responseHumedad = datos[2];
         }
         else
         {
            responseHora = "S/D";
            responseTemperatura = "S/D";
            responseHumedad = "S/D";
         }
         sleep(1000);

      } catch (IOException | TimeoutException | InterruptedException err) {
          err.printStackTrace();
      }
   }
}
```

Figura 18. Código de solicitud de temperatura

III. RESULTADOS

A. Servidor RPC (Llamada a Procedimiento Remoto) – Maqueta simulando vivienda real domótica

La función de la maqueta es simular una vivienda real a escala, esta contiene los componentes electrónicos que están conectados a los GPIO´s de la tarjeta Raspberry PI tal y como se mostró en la figura 19 de la sección II cada dispositivo funciona en base al mensaje enviado desde la interfaz cliente el cual cada dispositivo realiza una acción tal y como se describe a continuación:

- Tarjeta Raspberry PI modelo 3 B+: la función de esta es establecer puntos de conexión que permitan realizar una acción con cada elemento electrónico de entrada y salida (véase apéndice sección Raspberry PI).
- Switch de red: este dispositivo se usó con el fin de formar una red LAN y así establecer la comunicación entre la tarjeta Raspberry PI y la computadora donde se encuentra el servidor RabbitMQ y la interfaz de usuario.
- Sensor de gas MQ2: su función específica para la maqueta es detectar en el aire gases tóxicos que lleguen afectar a los habitantes del hogar, este sensor al momento de detectar algún gas toxico automáticamente se enviara una señal activando intermitentemente un buzzer al mismo tiempo es enviado un mensaje a la interfaz de usuario advirtiendo de dicho incidente. (véase apéndice sección Sensor MQ2)
- Sensor magnético: este tipo de sensor es utilizado para ser colocados en accesos a la vivienda, estos pueden representar dos estados posibles cerrado o abierto, al preguntar por dichos estados mediante los GPIO de la tarjeta Raspberry PI se procede a enviar una notificación según sea el estado de activación/desactivación de la alarma de intruso.
- Sensor de temperatura DHT11: el sensor de temperatura DHT11 mide también la humedad en el ambiente, la temperatura es solicitada de forma automática cuando se inicia la interfaz de usuario, el dato de temperatura se puede solicitar en diferentes escalas como Celsius, Fahrenheit, Kelvin y Rankine todas estas conversiones se realizan desde el programa servidor RPC (véase apéndice sección Sensor DHT11).
- Servo motor: tiene la función de simular un motor de paso para abrir o cerrar una puerta real, este servo motor está configurado para realice un giro establecido para indicar que se encuentra cerrado el acceso de igual forma se establece otro giro para indicar la apertura del acceso.
- Buzzer: este elemento solamente tiene la función de emitir un sonido cuando se hayan detectado anomalías en la vivienda como lo es la fuga de gases tóxicos o que intrusos intentaron saquear el hogar.

- Focos leds: estos componentes tienen la función de simular focos normales de una vivienda real, estos mediante la interfaz gráfica puede enviar a través del servidor RabbitMQ la (s) instrucciones encender, apagar o regular la luminosidad.

Es importante especificar que cuando la tarjeta Raspberry PI enciende e inicia el sistema operativo, mediante el programa Crontab se le especifico que ejecutará un script con el programa servidor RPC con el que estará interactuando el cliente [13].

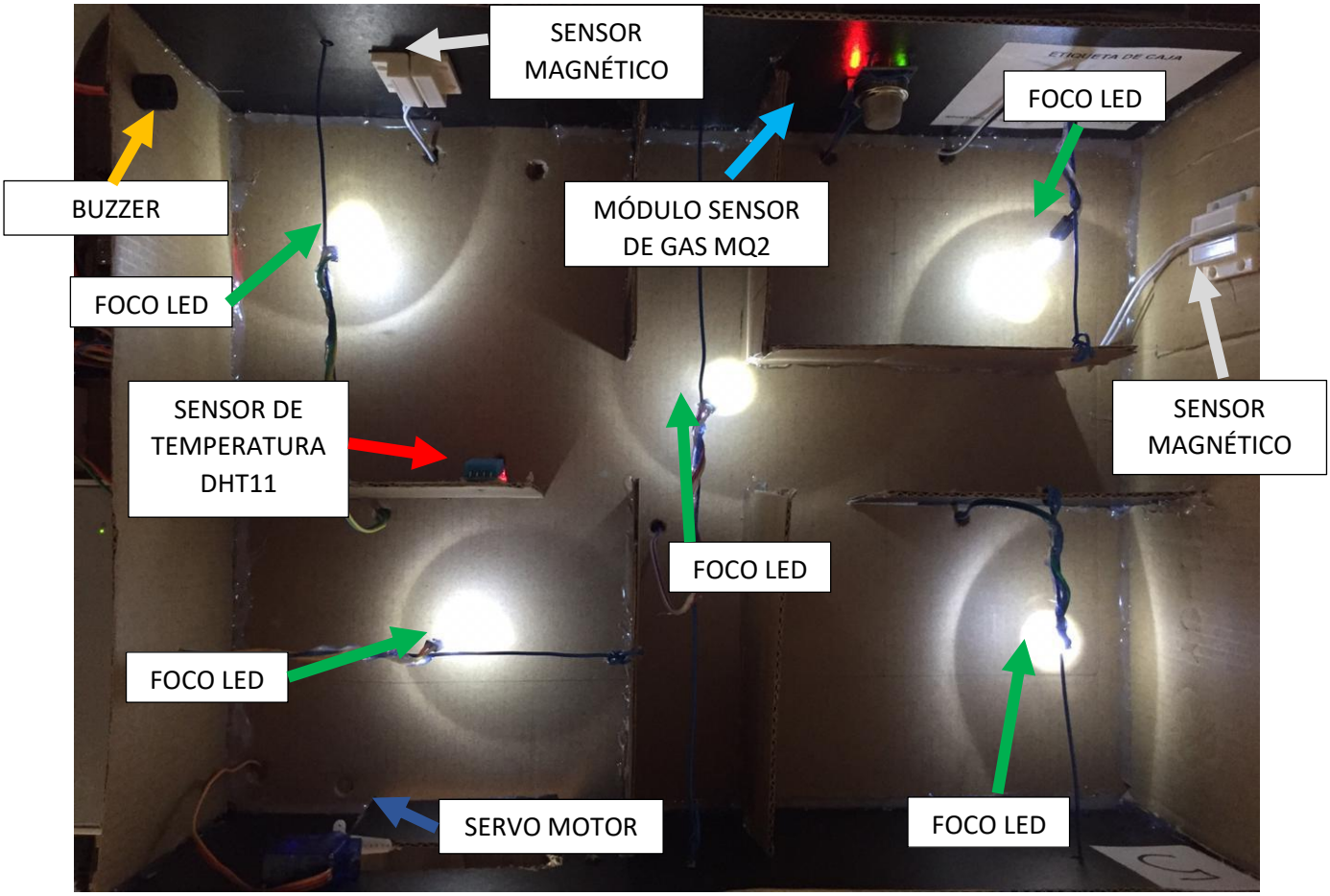

Figura 19. Interacción de los dispositivos electronicos a través de una maqueta simulando una vivienda domótica

B. Cliente RPC (Llamada a Procedimiento Remoto) – Interfaz de usuario

La interfaz de usuario se desarrolló bajo lenguaje de programación Java, esta es un panel de control que se visualiza en la figura 20 para la vivienda domótica para realizar las acciones que se describirán a continuación:

a) Sección 1 - 'CONTROL DE FOCOS LED': Esta sección tiene tres funciones para manipular el encendido, apagado y regulación de la intensidad de cada uno de los focos leds. Cuando se requiera encender uno o varios focos led en específico basta con que se seleccione la casilla de verificación y en automático el/los foco(s) led encenderán con una intensidad por default del 100% junto a ello se habilitara el

control deslizante que se encuentra en la parte inferior de cada casilla de verificación a fin de regular la intensidad el cual tiene un rango de luminosidad 0% a 100%. Adicionalmente, en este apartado se dispone de dos botones el cual su función es como su nombre lo describe son para encender todos los focos leds y otro para apagarlos en un momento dado.

b) Sección 2 – 'HISTORIAL DE SUCESOS': Este apartado tiene la función de mostrar una bitácora con los sucesos que han afectado a la vivienda los cuales se consideraron de dos tipos: detección de intrusos y fuga de gases tóxicos estas afectaciones si en un momento se requieren almacenar los datos del accidente, el usuario debe presionar el botón 'GUARDAR DATOS ACCIDENTES' en automático se actualizara el historial de sucesos.

c) Sección 3 – 'TEMPERATURA Y HUMEDAD': La temperatura y humedad está dada bajo el sensor DHT11 y estos parámetros son solicitados de forma automática y es mostrado en las cajas de texto de la sección, esta información se obtiene a través de la ejecución de un hilo de Java que solicita la información mediante el servidor RabbitMQ retornando de esta manera la fecha y hora de la solicitud y la temperatura en grados Celsius que es la escala de temperatura por default o bien existe una lista con diferentes escalas de temperatura como: grados Fahrenheit, Kelvin y Rankine en donde estas escalas son calculadas desde el servidor RPC partiendo de la temperatura en grados Celsius adicionalmente retorna en porcentaje la humedad que hay en el ambiente.

d) Sección 4 – 'NOTIFICACIÓN DE INCENDIO': En la vivienda se colocó un sensor de gas MQ-2, el cual cuando se detecta en el ambiente algún gas considerado tóxico en automático se visualiza un mensaje a la interfaz de usuario advirtiendo de alguna fuga de gas peligroso además de la fecha y hora de detección, es importante mencionar que mientras exista una fuga de gas la hora se estará actualizando visualizando así en las cajas de texto de la sección la última actualización de detección de gas. Este proceso se ejecuta de forma constante para mantener al usuario pendiente de esta anomalía.

e) Sección 5 – 'NOTIFICACIÓN DE INTRUSO': La vivienda posee un par de sensores magnéticos ubicados respectivamente en la puerta y otro en la ventana, al momento de activar la alarma en el hogar se solicita al usuario que estos dos accesos se encuentren cerrados de lo contrario se mostrara una advertencia de la no activación de la alarma. Cuando se activa la alarma mostrara un mensaje que la alarma está a la espera de que los sensores detecten algún acceso abierto si ocurre dicha actividad en automático el buzzer emitirá un sonido y en la interfaz gráfica mostrara la fecha y hora, así como el acceso por donde saquearon la vivienda.

f) Sección 6 – 'CONTROL DE PUERTA': Los botones de esta sección tienen dos funciones básicas abrir o cerrar la puerta, cabe señalar que cuando la alarma de intruso se activa por el usuario la puerta en caso de estar abierta se cierra en automático y enseguida los dos botones con la funcionalidad mencionada se inhabilitan hasta que el usuario desactive la alarma de intruso.

g) Sección 7 – ‘ACTIVACIÓN Y DESACTIVACIÓN DE ALARMA’: El botón para desactivar la alarma sirve para desactivar el dispositivo buzzer colocado en la vivienda, como se sabe en la interfaz se colocaron dos tipos de notificaciones que afectan a la vivienda (incendio o intruso) cualquier alteración notificada inmediatamente el buzzer comienza a emitir un sonido advirtiendo que ocurrió algún accidente por lo tanto para desactivar el sonido basta con presionar el botón de desactivar alarma.

Figura 20. Interfaz de usuario cliente

IV. CONCLUSIONES

El presente trabajo se realizo el diseño y desarrolló de forma satisfactoria las funciones básicas de una vivienda domótica, estas son manejadas a través de la interfaz de usuario que se desarrolló para que el cliente interactúe de forma amigable con su hogar. Para proyecciones futuras este diseño puede a escalar agregando más funciones de interactividad que cubran más necesidades del cliente para brindar mayor confort al cliente.

V. RECOMENDACIONES

El presente prototipo desarrollado trata de simular una vivienda domótica a través de los elementos electrónicos básicos, por lo tanto, si se requiere implementar este proyecto en una vivienda real lo que se requeriría es básicamente implementar la parte eléctrica y electrónica adicional que ocuparía con cada uno de los dispositivos electrónicos de entrada y salida de la vivienda para que estos de igual manera puedan ser manipulados a través de los GPIO de la tarjeta Raspberry PI. Adicionalmente a esto la interfaz de control de los

elementos de la vivienda que se desarrolló en Java puede escalar a ser una aplicación móvil para que sea más sencillo y practico la interacción con todos los elementos de la vivienda domótica. Asociado a esto una vivienda domótica puede contener más dispositivos que brinden mayor comodidad para los usuarios, en el proyecto solo se abordaron las funciones básicas de domótica. Por otro lado, el utilizar el servidor RabbitMQ como intermediario para la recepción y envío de mensajes a través de las queue´s están diseñadas para proporcionar un transporte ligero de la publicación y consumo de mensajes.

Por ejemplo, considerando el sensor de temperatura que mide la temperatura y humedad se desarrolló una función que envía la temperatura a una cola de mensajes cada vez que hay un cambio en la temperatura en lugar de estar procesando los datos en la vivienda de forma manual y estar conectado todo el tiempo para realizar dicho censo, en lugar de mantener una aplicación que procesa los datos, simplemente envía los datos y maneja los datos en otro servicio. Esto requiere menos ancho de banda de red.

Por lo tanto, una queue de mensajes permite que su aplicación tenga un bajo consumo de energía, envíe paquetes de datos minimizados y distribuya información de manera eficiente a uno o varios receptores partiendo del servidor RabbitMQ.

VI. APÉNDICES

Datos de especificación de los sensores

A. Sensor de gas MQ-2

¿Cómo funciona el sensor MQ-2?

Cuando el dióxido de estaño (partículas semiconductoras) se calienta en el aire a alta temperatura, el oxígeno se adsorbe en la superficie. En el aire limpio, los electrones donados en el dióxido de estaño son atraídos hacia el oxígeno que se adsorbe en la superficie del material sensor. Esto evita el flujo de corriente eléctrica. En presencia de gases reductores, la densidad superficial del oxígeno adsorbido disminuye a medida que reacciona con los gases reductores. Luego se liberan electrones en el dióxido de estaño, permitiendo que la corriente fluya libremente a través del sensor.

El voltaje de salida analógico proporcionado por el sensor cambia proporcionalmente a la concentración de humo/gas. Cuanto mayor es la concentración de gas, mayor es el voltaje de salida; mientras que una menor concentración de gas produce un bajo voltaje de salida. La señal analógica del sensor de gas MQ2 se alimenta aún más al comparador de alta precisión LM393 (soldado en la parte inferior del módulo), por supuesto, para digitalizar la señal. Junto con el comparador hay un pequeño potenciómetro que puede girar para ajustar la sensibilidad del sensor. Se puede usar para ajustar la concentración de gas a la que el sensor lo detecta.

Características del MQ-2 Sensor de Gas:

- Voltaje de Operación adecuado: 5V DC
- Respuesta rápida y alta sensibilidad

- Rango de detección: 300 a 10000 ppm
- Tiempo de Respuesta: ≤ 10s
- Tiempo de recuperación: ≤ 30s
- Temperatura de trabajo: -20 ℃ - +55 ℃
- Consume menos de 150mA a 5V [15].

B. Sensor de temperatura DHT11

El DHT11 es un sensor de temperatura y humedad digital de bajo costo. Utiliza un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no hay pines de entrada analógica). Es bastante simple de usar, pero requiere sincronización cuidadosa para tomar datos.

Características

- Alimentación: 3Vdc ≤ Vcc ≤ 5Vdc
- Rango de medición de temperatura: 0 a 50 °C
- Precisión de medición de temperatura: ±2.0 °C .
- Rango de medición de humedad: 20% a 90% RH.
- Precisión de medición de humedad: 4% RH.
- Tiempo de censado: 1 seg [16].

C. Sensor magnético

El Sensor magnético para ventanas y puertas consta de un imán y un reedswitch (interruptor magnético). Este sensor funciona como un switch normalmente abierto (mientras hay campo magnético). Cuando la puerta o ventana se abre, el circuito eléctrico también se cierra y es posible detectar la apertura de la misma. El sensor viene completamente sellado en plástico lo que lo hace extremadamente resistente [17,18].

VII. REFERENCIAS

[1] "Domótica en el hogar: ¿Qué es? ¿Cómo funciona? - Casas Inteligentes", Casasdomoticass.com, 2019. [Online]. Available: https://casasdomoticass.com/domotica/. [Accessed: 22- Nov- 2019].

[2] "Qué entendemos por casa domótica", Casas Digitales, 2019. [Online]. Available: https://www.casasdigitales.com/que-es-casa-domotica/. [Accessed: 23- Nov- 2019].

[3] 2019. [Online]. Available: https://www.researchgate.net/publication/262670875_Smart_Domotic_houses. [Accessed: 22- Nov- 2019].

[4] "RabbitMQ tutorial - Remote procedure call (RPC) — RabbitMQ", Rabbitmq.com, 2019. [Online]. Available: https://www.rabbitmq.com/tutorials/tutorial-six-python.html. [Accessed: 23- Nov- 2019].

[5] "RPIO", PyPI, 2019. [Online]. Available: https://pypi.org/project/RPIO/. [Accessed: 23- Nov- 2019].

[7] "Qué es el protocolo SSL/TLS | Redalia", Redalia.es, 2019. [Online]. Available: https://www.redalia.es/ssl/protocolo-ssl/. [Accessed: 23- Nov- 2019].

[8] R. Blog and R. Blog, "Que es BOARD y BCM en Raspberry Pi", Mnp.cl, 2019. [Online]. Available: https://www.mnp.cl/post/que-es-board-bcm-raspberry-pi. [Accessed: 23- Nov- 2019].

[9]" ¿Qué es PWM? - Control de GPIO con Python en Raspberry Pi", https://www.programoergosum.com, 2019. [Online]. Available: https://www.programoergosum.com/cursos-online/raspberry-pi/238-control-de-gpio-con-python-en-raspberry-pi/que-es-pwm. [Accessed: 23- Nov- 2019].

[10] "TLS Support — RabbitMQ", *Rabbitmq.com*, 2019. [Online]. Available: https://www.rabbitmq.com/ssl.html. [Accessed: 23- Nov- 2019].

[11] "Fórmulas de conversión de temperatura", Elosiodelosantos.com, 2019. [Online]. Available: http://www.elosiodelosantos.com/sergiman/div/formulas_conversion_de_temperaturas.htm. [Accessed: 23- Nov- 2019].

[12] "Why message queues for IoT projects? - CloudAMQP", Cloudamqp.com, 2019. [Online]. Available: https://www.cloudamqp.com/blog/2017-05-13-why-message-queues-for-iot-projects.html. [Accessed: 23- Nov- 2019].

[13] "Cómo utilizar Cron y Crontab en Linux para programar tareas", Redeszone.net, 2019. [Online]. Available: https://www.redeszone.net/2017/01/09/utilizar-cron-crontab-linux-programar-tareas/. [Accessed: 23- Nov- 2019].

[14] "SSL - Python Wiki", Wiki.python.org, 2019. [Online]. Available: https://wiki.python.org/moin/SSL. [Accessed: 23- Nov- 2019].

[15] "MQ-2 Sensor de Gas - Geek Factory", Geek Factory, 2019. [Online]. Available: https://www.geekfactory.mx/tienda/sensores/mq-2-sensor-de-gas/. [Accessed: 23- Nov- 2019].

[16] "Sensor de Temperatura y Humedad DHT11 - Electronilab", Electronilab, 2019. [Online]. Available: https://electronilab.co/tienda/sensor-de-temperatura-y-humedad-dht11/. [Accessed: 23- Nov- 2019].

[17] "Sensor magnético para puertas y ventanas - Geek Factory", Geek Factory, 2019. [Online]. Available: https://www.geekfactory.mx/tienda/sensores/sensor-magnetico-para-ventanas-y-puertas/. [Accessed: 23- Nov- 2019].

[18] "sensor magnético – erizorunner", erizorunner, 2019. [Online]. Available: https://erizorunner.wordpress.com/tag/sensor-magnetico/. [Accessed: 23- Nov- 2019].