



# Control de Sensores y Periféricos mediante servicios API-REST

Tecnología de Software para  
Electrónica

## Trabajo de Investigación

Autor:

Jordy Bayas Escobar (alumno)

Joshua Reyes Jurado (alumno)

Roger Espinoza Naranjo (alumno)

Tutor/es:

Darwin Alulema Flores (tutor1)

12 de Junio de 2019





# Control de Sensores y Periféricos mediante servicios API-REST

---

Subtítulo del proyecto

## Autor/es

Jordy Bayas Escobar (alumno)  
Joshua Reyes Jurado (alumno)  
Roger Espinoza Naranjo (alumno)

## Tutor/es

Darwin Alulema Flores (tutor1)  
*Departamento de Eléctrica y Electrónica*

Tecnología de Software para Electrónica



Quito, 12 de Junio de 2019



# Índice general

<b>1</b>	<b>Planteamiento del Problema</b>	<b>1</b>
<b>2</b>	<b>Objetivos</b>	<b>3</b>
2.1	Objetivo General . . . . .	3
2.1.1	Objetivos específicos . . . . .	3
<b>3</b>	<b>Estado del Arte</b>	<b>5</b>
<b>4</b>	<b>Marco Teórico</b>	<b>9</b>
4.1	Servicio API-REST . . . . .	9
4.1.1	Definición . . . . .	9
4.1.2	Ejemplo . . . . .	9
4.1.3	Requisitos clave para una API REST . . . . .	9
4.2	MySQL . . . . .	10
4.3	Spring Boot . . . . .	10
4.4	NODEMCU . . . . .	12
4.5	JSON . . . . .	12
<b>5</b>	<b>Diagramas</b>	<b>15</b>
<b>6</b>	<b>Lista de Componentes</b>	<b>17</b>
6.1	Servicios API-REST . . . . .	17
6.2	ESP8266 . . . . .	18
6.3	Arduino IDE . . . . .	18
6.4	Sensor DHT11 . . . . .	18
6.5	Relé . . . . .	19
6.6	Sensor Ultrasónico HC-SR04 . . . . .	19
6.7	LCD . . . . .	19
<b>7</b>	<b>Mapa de Variables</b>	<b>21</b>
<b>8</b>	<b>Explicación</b>	<b>23</b>
<b>9</b>	<b>Descripción de Prerrequisitos y Configuración</b>	<b>27</b>
9.0.1	Arduino IDE . . . . .	27
9.0.2	Eclipse . . . . .	27

9.0.3 Librería NodeMCu . . . . .	27
<b>10 Aportaciones</b>	<b>29</b>
<b>11 Conclusiones</b>	<b>31</b>
<b>12 Recomendaciones</b>	<b>33</b>
<b>13 Cronograma</b>	<b>35</b>
<b>14 Repositorio</b>	<b>37</b>
14.1 Git Hub . . . . .	37
14.2 Links Overleaf . . . . .	37
<b>15 Anexos</b>	<b>39</b>
15.1 Manual de usuario . . . . .	39
<b>Bibliografía</b>	<b>41</b>

---

# Índice de figuras

4.1	API-REST . . . . .	10
4.2	API-REST . . . . .	11
4.3	Spring Boot . . . . .	11
4.4	NODEMCU . . . . .	12
4.5	JSON . . . . .	13
5.1	Servicios REST . . . . .	15
5.2	Comunicaciones . . . . .	15
6.1	API-REST . . . . .	17
6.2	NODEMCU ESP8266 . . . . .	18
6.3	Sensor DHT11 . . . . .	18
6.4	Relé ELectrónico . . . . .	19
6.5	HC-sr04 . . . . .	19
6.6	LCD . . . . .	20
7.1	Conexión con los servicios API-REST . . . . .	21
8.1	Vista de base de datos . . . . .	25
8.2	Código en Eclipse . . . . .	25
13.1	Cronograma de actividades parte 1 . . . . .	35
13.2	Cronograma de actividades parte 2 . . . . .	35
15.1	Librería ESP . . . . .	39
15.2	Librería ESP . . . . .	39
15.3	Librería ESP . . . . .	40





# 1 Planteamiento del Problema

## **Control de Sensores y Periféricos mediante servicios API-REST**

A medida que transcurre el tiempo, ha venido en desarrollo la utilización del IOT, el cual cabe destacar que trata de utilizar las cosas cotidianas, mediante el uso de servicios de consumo en internet, conforme al utilizar estas funciones también incentivan a la creación de los mismos, es por ello que conforme se desarrollaban realizaban modelos que pueda realizar tanto la medición de sensores para poder detectar las variaciones que se puede producir en un entorno, y actuadores que se pueden activar dependiendo de las necesidades del usuario es por ello se debe poder administrar de manera eficiente los datos y las funciones implementadas para los actuadores, y los datos controlados por el usuario.



## **2 Objetivos**

### **2.1 Objetivo General**

Desarrollar un sistema basado en API-REST que permita control de distintos sensores y actuadores

#### **2.1.1 Objetivos específicos**

- Especificar las variables para el desarrollo del sistema para los distintos sensores y actuadores.
- Identificar los distintos sensores de Humedad, Temperatura y de proximidad a controlar mediante el ESP8266 y el servicio API-REST.
- Deducir las distintas aplicaciones mediante la utilización del módulo WI-FI ESP-8266.
- Diseñar un prototipo basado en API-REST para que cada uno de los sensores esté asociado a un actuador predeterminado.



### 3 Estado del Arte

Al revisar distintos ambitos en donde se esta llegando a aplicar los distintos módulos, se puede encontrar grandes temas, por el cual se llegó al artículo realizado por J.Chandramohan, R.Nagarajan, K. Satheeshkumar, N.Ajithkumar, A.Gopinath, S.Ranjithkumar los cuales en su paper " Intelligent Smart Home Automation and Security System Using Arduino and Wi-fi" donde se resume una aplicación de un sistema de control y monitoreo del hogar flexible y de bajo costo con la ayuda de un equipo integrado.[1]

Servidor de micro-web con conectividad de protocolo de Internet (IP) para acceder y controlar equipos y dispositivos de forma remota mediante aplicación para smartphone basada en Android, y utilizando Arduino, como evidencia se puede aseverar que en la actualidad hay diversas apelaciones que se vienen realizando estos procedimientos, además de incluir una base de datos donde se guarda todo los cambios que has realizado en los equipos, pero además hay otro artículos donde se realizan mediante conexiones LAN como es el caso de "Home automation using arduino WiFi module ESP8266" de Kotiyal Bandanawaz, Baig Iliyas, Muzamil Muzamil Chiktay, Dalv Salahuddin donde ellos proponen un enfoque diferente donde se presenta un diseño y una implementación prototipo de un nuevo sistema de automatización del hogar que utiliza la tecnología Wi-Fi como una infraestructura de red que conecta sus partes. Donde consta de un servidor (servidor web), que presenta el núcleo del sistema que administra, controla y supervisa la casa de los usuarios el cual se puede administrar nivel local (LAN) o de forma remota (internet). La segunda parte es el módulo de interfaz de hardware, que proporciona una interfaz adecuada para los sensores y el actuador del sistema doméstico. Esto llega a ser interesante, porque se puede intuir 2 cosas hoy en día ya no se es limitado por una red de área local, es decir ahora se puede controlar ingresando directamente vía internet a un servidor web específico que controle distintas funciones, en varios ámbitos.[2]

Otro artículo "IoT based smart water tank with Android application" hecho por Priyen P. Shah, Anjali A. Patil, Subodh S. Ingleshwar, el cual menciona otro uso mediante aplicaciones esta vez tratándose monitoreos contantes del nivel del agua, pero usando sensores, la señal de estos sensores es captada por el arduino y transmitida por el módulo wi-fi para poder receptar mediante la aplicación basada en android. Este paper abarca una propuesta diferente utilizando el internet de las cosas, que permite la interacción con aplicaciones externas, como la interfaz ofrecida por los componentes individuales de la infraestructura, que permite la interacción y la coordinación entre los componentes. Básicamente su base viene dada para contribuir a una proyección

estándar de las mediciones y control del agua dentro de tanques además eliminar cualquier inconveniente y proporcionar una solución eficiente y económica.[3]

Al observar el artículo "Intelligent multi-sensor control system based on innovative technology integration via ZigBee and Wi-Fi networks" de los autores Kuang-Yow Lian, Sung-Jung Hsiao y Wen-Tsai Sung, donde se describe que una red de transmisión de datos es uno de los problemas más difíciles de resolver en los sistemas de redes de sensores inalámbricos. ZigBee y Wi-Fi pertenecen a diferentes protocolos de red. Si un sistema de red debe usar ZigBee y Wi-Fi al mismo tiempo para transmitir datos, se presenta un desafío considerable. Este artículo presenta un nuevo método de hardware que integra ZigBee y Wi-Fi. El método propuesto se basa en el módulo portátil de Arduino ZigBee y el concepto de Ethernet. Este estudio construye un sistema inteligente de control de electrodomésticos utilizando la red ZigBee. Este sistema de control inteligente utiliza una arquitectura de red integrada ZigBee y Wi-Fi en la casa. Nuestro estudio envía los mensajes del sensor ZigBee a una base de datos en la nube a través de la red de protocolo TCP / IP que contiene la red física y las líneas de dispositivos de red inalámbrica. El control de acceso a la gestión se logra mediante teléfonos inteligentes. El método propuesto es muy simple y fácil de implementar utilizando circuitos Arduino. La efectividad del método propuesto se verifica mediante la simulación y los resultados experimentales. Los componentes de hardware incluyen el controlador Arduino, el módulo de comunicación inalámbrica XBee Serie 2 y los sensores del dispositivo final. Los lenguajes de programación de Android y Java. Se utilizan para escribir el teléfono inteligente y los programas de reconocimiento del servidor.[? ]

Viendo otros usos que se pueden dar las utilidades del módulo se puede encontrar el artículo "Taking Arduino to the Internet of Things: The ASIP programming model" realizado por Gianluca Barbona, Michael Margolis, Filippo Palumbo, Franco Raimondi, Nick Weldin nos menciona que los microcontroladores como Arduino son ampliamente utilizados por todo tipo de fabricantes en todo el mundo. La popularidad se debe a la simplicidad de uso de Arduino y la gran cantidad de sensores y bibliotecas disponibles para ampliar las capacidades básicas de estos controladores. La última década ha sido testigo de una oleada de soluciones de ingeniería de software para "Internet of Things", pero en varios casos estas soluciones requieren recursos computacionales que son más avanzados que los microcontroladores simples y con recursos limitados. En el presente documento presentan el modelo de Programación de Interfaz de Servicio de Arduino (ASIP), un nuevo modelo que aborda los problemas anteriores proporcionando una abstracción de "Servicio" para agregar fácilmente nuevas capacidades a los microcontroladores, y brindando soporte para tableros en red que utilizan una variedad de estrategias, que incluyen conexiones de socket, dispositivos de puente, publicación basada en MQTT, mensajería de suscripción, servicios de descubrimiento, etc. Ofrecemos una implementación de código abierto del código que se ejecuta en las placas Arduino y las bibliotecas de clientes en Java, Python, Racket y Erlang. Mostramos cómo ASIP permite el rápido desarrollo de aplicaciones no triviales (coordinación de entrada / sa-

---

lida en tableros distribuidos e implementación de un algoritmo de seguimiento de línea para un robot remoto) y evaluamos el rendimiento de ASIP de varias maneras, tanto cuantitativas como cualitativas. [4]

Como se puede observar el tema de la investigación tiene diversas investigaciones, artículos que poseen cierta relación con lo que se está investigando, pero este trabajo se diferencia de los demás en que se quiere integrar el módulo ESP8266 de manera que interactue con el servicio API-REST mediante la conexión WI-FI. Se profundizará en varios temas que no se han investigado y se orientara más a las distintas ventajas que ofrece esta clase de servicios, para poder ser aplicado utilizando los distintos actuadores. Por lo que nuestro trabajo unirá los beneficios de plataformas gratuitas que ofrezcan servicios especiales con las características del ESP8266, los servicios API-REST y además de una aplicación para demostrar el potencial que puede tener el uso de estos.

---





## 4 Marco Teórico

### 4.1 Servicio API-REST

#### 4.1.1 Definición

Una API REST define un conjunto de funciones que los desarrolladores pueden realizar solicitudes y recibir respuestas a través del protocolo HTTP, como GET y POST.

Debido a que la API REST usa HTTP, pueden ser utilizados por prácticamente cualquier lenguaje de programación y son fáciles de probar (es un requisito de una API REST que el cliente y el servidor sean independientes entre sí, lo que permite codificarlo en cualquier idioma y mejorar al soportar la longevidad y evolución).

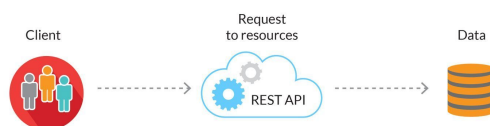
#### 4.1.2 Ejemplo

Twitter proporciona una API REST que puede consultar para obtener los últimos tweets, puede proporcionar una consulta de búsqueda (o una etiqueta hash) y devolverá los resultados en formato JSON. Ejemplo de esta solicitud HTTP a la API de Twitter para obtener los últimos 3 tweets que coinciden con "jQuery".

#### 4.1.3 Requisitos clave para una API REST

- Debe usar estándares web donde tengan sentido ser amigable para el desarrollador y ser explorable a través de una barra de direcciones del navegador
- Debe ser simple, intuitivo y consistente para hacer que la adopción no solo sea fácil sino también agradable
- Debe proporcionar suficiente flexibilidad para impulsar la mayoría de la interfaz de usuario de encantamiento. Debe ser eficiente, manteniendo el equilibrio con los otros requisitos
- 

Una API es la interfaz de usuario de un desarrollador: al igual que cualquier UI, es importante asegurarse de que la experiencia del usuario se piense cuidadosamente.



**Figura 4.1:** API-REST

## 4.2 MySQL

Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo,<sup>12</sup> y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

MySQL fue inicialmente desarrollado por MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius). MySQL AB fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por Oracle Corporation en 2010, la cual ya era dueña desde 2005 de InnoDB Oy, empresa finlandesa desarrolladora del motor InnoDB para MySQL.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de doble licenciamiento anteriormente mencionado. La base de datos se distribuye en varias versiones, una Community, distribuida bajo la Licencia pública general de GNU, versión 2, y varias versiones Enterprise, para aquellas empresas que quieran incorporarlo en productos privativos. Las versiones Enterprise incluyen productos o servicios adicionales tales como herramientas de monitorización y asistencia técnica oficial. En 2009 se creó un fork denominado MariaDB por algunos desarrolladores (incluido algunos desarrolladores originales de MySQL) descontentos con el modelo de desarrollo y el hecho de que una misma empresa controle a la vez los productos MySQL y Oracle.

## 4.3 Spring Boot

Es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.

La primera versión fue escrita por Rod Johnson, quien lo lanzó junto a la publicación de su libro *Expert One-on-One J2EE Design and Development* (Wrox Press, octubre



**Figura 4.2:** API-REST

2002). El framework fue lanzado inicialmente bajo la licencia Apache 2.0 en junio de 2003. El primer gran lanzamiento fue la versión 1.0, que apareció en marzo de 2004 y fue seguida por otros hitos en septiembre de 2004 y marzo de 2005. La versión 1.2.6 de Spring Framework obtuvo reconocimientos Jolt Awards y Jax Innovation Awards en 2006. Spring Framework 2.0 fue lanzada en 2006, la versión 2.5 en noviembre de 2007, Spring 3.0 en diciembre de 2009, y Spring 3.1 dos años más tarde. El inicio del desarrollo de la versión 4.0 fue anunciado en enero de 2013. La versión actual es la 5.0.

Si bien las características fundamentales de Spring Framework pueden ser usadas en cualquier aplicación desarrollada en Java, existen variadas extensiones para la construcción de aplicaciones web sobre la plataforma Java EE. A pesar de que no impone ningún modelo de programación en particular, este framework se ha vuelto popular en la comunidad al ser considerado una alternativa, sustituto, e incluso un complemento al modelo EJB (Enterprise JavaBean).



**Figura 4.3:** Spring Boot

## 4.4 NODEMCU

Es una plataforma IoT de código abierto. Incluye el firmware que se ejecuta en el SoC Wi-Fi ESP8266 de Espressif Systems y el hardware que se basa en el módulo ESP-12. El término "NodeMCU" se refiere al firmware en lugar de a los kits de desarrollo. El firmware utiliza el lenguaje Lua. Se basa en el proyecto eLua y se basa en el SDK no operativo de Espressif para el ESP8266. Utiliza muchos proyectos de código abierto, como lua-cjson, y spiffs.



**Figura 4.4:** NODEMCU

## 4.5 JSON

Es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera (año 2019) un formato independiente del lenguaje.

Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos es que resulta mucho más sencillo escribir un analizador sintáctico (parser) para él. En JavaScript, un texto JSON se puede analizar fácilmente usando la función `eval()`, algo que (debido a la ubicuidad de JavaScript en casi cualquier navegador web) ha sido fundamental para que haya sido aceptado por parte de la comunidad de desarrolladores AJAX.

En la práctica, los argumentos a favor de la facilidad de desarrollo de analizadores

---

o de sus rendimientos son poco relevantes, debido a las cuestiones de seguridad que plantea el uso de `eval()` y el auge del procesamiento nativo de XML incorporado en los navegadores modernos. Por esa razón, JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia (de aquí su uso por Yahoo!, Google, Mozilla, etc, que atienden a millones de usuarios) cuando la fuente de datos es explícitamente de fiar y donde no es importante el hecho de no disponer de procesamiento XSLT para manipular los datos en el cliente.



**Figura 4.5:** JSON



## 5 Diagramas

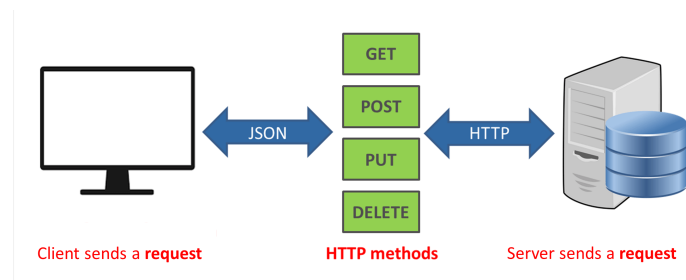


Figura 5.1: Servicios REST

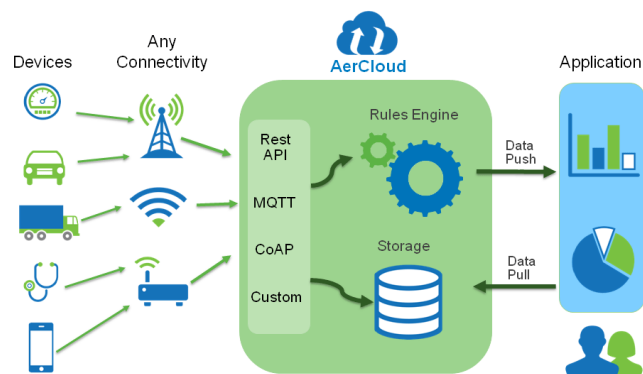


Figura 5.2: Comunicaciones





# 6 Lista de Componentes

## 6.1 Servicios API-REST

API REST debe a que es un estándar lógico y eficiente para la creación de servicios web. Por poner algún ejemplo tenemos los sistemas de identificación de Facebook, la autenticación en los servicios de Google (hojas de cálculo, Google Analytics). Según Fielding las restricciones que definen a un sistema RESTful serían:

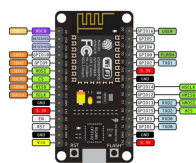
- **Cliente-servidor:** esta restricción mantiene al cliente y al servidor débilmente acoplados. Esto quiere decir que el cliente no necesita conocer los detalles de implementación del servidor y el servidor se “despreocupa” de cómo son usados los datos que envía al cliente.
- **Sin estado:** aquí decimos que cada petición que recibe el servidor debería ser independiente, es decir, no es necesario mantener sesiones.
- **Cacheable:** debe admitir un sistema de almacenamiento en caché. La infraestructura de red debe soportar una caché de varios niveles. Este almacenamiento evitará repetir varias conexiones entre el servidor y el cliente para recuperar un mismo recurso.
- **Interfaz uniforme:** define una interfaz genérica para administrar cada interacción que se produzca entre el cliente y el servidor de manera uniforme, lo cual simplifica y separa la arquitectura. Esta restricción indica que cada recurso del servicio REST debe tener una única dirección, “URI”.
- **Sistema de capas:** el servidor puede disponer de varias capas para su implementación. Esto ayuda a mejorar la escalabilidad, el rendimiento y la seguridad.



Figura 6.1: API-REST

## 6.2 ESP8266

Es una plataforma IoT de código abierto. Incluye el firmware que se ejecuta en el SoC Wi-Fi ESP8266 de Espressif Systems y el hardware que se basa en el módulo ESP-12. El término "NodeMCU" se refiere al firmware en lugar de a los kits de desarrollo. El firmware utiliza el lenguaje Lua. Se basa en el proyecto eLua y se basa en el SDK no operativo de Espressif para el ESP8266. Utiliza muchos proyectos de código abierto, como lua-cjson, y spiffs.<sup>5</sup>



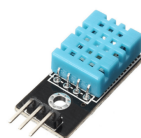
**Figura 6.2:** NODEMCU ESP8266

## 6.3 Arduino IDE

Es una plataforma electrónica de código abierto basada en hardware y software fáciles de usar. Las placas Arduino pueden leer entradas (luz en un sensor, un dedo en un botón o un mensaje de Twitter) y convertirla en una salida: activar un motor, encender un LED y publicar algo en línea. Puede decirle a su tarjeta qué debe hacer enviando un conjunto de instrucciones al microcontrolador de la tarjeta. Para hacerlo, utiliza el lenguaje de programación Arduino (basado en Wiring ) y el software Arduino (IDE) , basado en el procesamiento .

## 6.4 Sensor DHT11

El sensor DHT11 que nos permite medir la temperatura y humedad con Arduino. Una de las ventajas que nos ofrece el DHT11, además de medir la temperatura y la humedad, es que es digital. A diferencia de sensores como el LM35, este sensor utiliza un pin digital para enviarnos la información y por lo tanto, estaremos más protegidos frente al ruido.

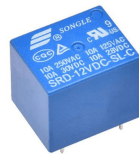


**Figura 6.3:** Sensor DHT11

---

## 6.5 Relé

Es un dispositivo electromagnético. Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes.



ElectroCrea.com

**Figura 6.4:** Relé ELectrónico

## 6.6 Sensor Ultrasónico HC-SR04

El HC-SR04 es un sensor de distancias por ultrasonidos capaz de detectar objetos y calcular la distancia a la que se encuentra en un rango de 2 a 450 cm.



**Figura 6.5:** HC-sr04

## 6.7 LCD

Es una pantalla delgada y plana formada por un número de píxeles en color o monocromos colocados delante de una fuente de luz o reflectora. A menudo se utiliza en dispositivos electrónicos de pilas, ya que utiliza cantidades muy pequeñas de energía eléctrica.



**Figura 6.6:** LCD

## 7 Mapa de Variables



Figura 7.1: Conexión con los servicios API-REST



## 8 Explicación

En base a lo que se puede utilizar, los distintos códigos se obtienen primeramente, debemos crear métodos get y post para cada uno de los nodemcu, por ello también se necesita extraer los valores de los sensores y actualizar de manera real los valores de los mismos, además de poder interactuar con los sensores.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include<ESP8266WiFi.h>
#include<ESP8266HTTPClient.h>

//Conexion WiFi
const char* ssid = "Rogeri7";
const char* password = "11223348";

String url1 = "http://172.20.10.5:8080/notes/1";
String url2 = "http://172.20.10.5:8080/notes/";

//Variables Pantalla LCD
LiquidCrystal_I2C lcd(0x27,20,4);
int Promedio=0,i=0,cont=0;

double tiempoDisplay=0,t=0;

void setup() {
  pinMode(0,OUTPUT);
  pinMode(4,OUTPUT);

  lcd.init();
  lcd.backlight();
  lcd.setCursor(10,0);
  Serial.begin(9600);
  //Promedio=ambiente(i);
  lcd.print("Dif.tp:");
```

---

```

    delay(10);

    // Conectar wifi
    WiFi.begin(ssid, password);
    Serial.println("no");
    while (WiFi.status() != WL_CONNECTED){
        delay(5000);
        Serial.println("R");
    }
    Serial.println("Red conectada");
    cont = 100;
}

void loop() {
    Serial.print("gggggg");
    HTTPClient http;
    cont = cont + 1;
    //Metodo Post
    if (http.begin(url2)) //Iniciar conexion
    {

        Serial.println("Entramos a la base");
        String JSON = "{\"id\":\"0\",\"title\":\"a\",\"contents\":\"d\",\"createAt\":\"0000-00-00T00:00:00Z\"}";
        //String JSON = "{\"name\":\"Alice\", \"age\":20,\"ssn\":\"123456789\",\"employedId\":\"00000000000000000000000000000000\"}";

        int httpCode = http.POST(JSON); //Realizar peticion
        Serial.print("[HTTP] POST...\n"); //Realizar pericion
        Serial.println(JSON);
        if (httpCode > 0) {
            Serial.printf("[HTTP] POST... code: %d\n", httpCode);

            if (httpCode == HTTP_CODE_OK || httpCode == HTTP_CODE_MOVED_PERMANENTLY) {
                String payload = http.getString(); //Obtener respuesta dato=http.getString()
                String aa=payload.
                Serial.print("Dato recibido:");
                Serial.println(payload); //Mostrar respuesta por serial
            }
        } else {
            Serial.printf("[HTTP] GET...failed, error:%s\n", http.errorToString(httpCode).c_str());
        }
    }
}

```

---

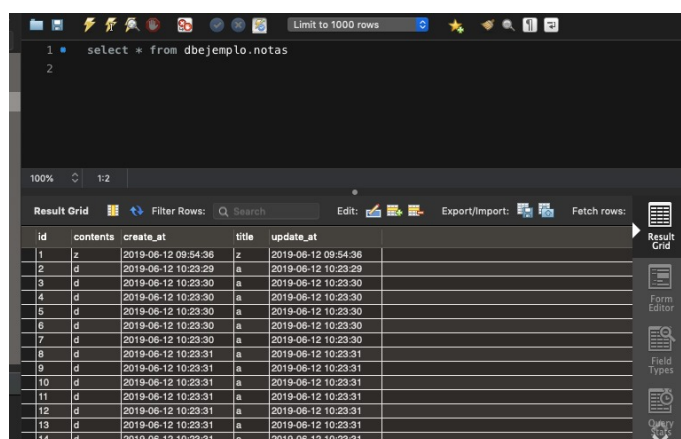


```

    }
    http.end();
} else
    delay(30000);

```

Se recibirá en la base de datos lo siguiente:// También, tendremos que declarar los



id	contents	create_at	title	update_at
1	z	2019-06-12 09:54:36	z	2019-06-12 09:54:36
2	d	2019-06-12 10:23:29	a	2019-06-12 10:23:29
3	d	2019-06-12 10:23:30	a	2019-06-12 10:23:30
4	d	2019-06-12 10:23:30	a	2019-06-12 10:23:30
5	d	2019-06-12 10:23:30	a	2019-06-12 10:23:30
6	d	2019-06-12 10:23:30	a	2019-06-12 10:23:30
7	d	2019-06-12 10:23:30	a	2019-06-12 10:23:30
8	d	2019-06-12 10:23:31	a	2019-06-12 10:23:31
9	d	2019-06-12 10:23:31	a	2019-06-12 10:23:31
10	d	2019-06-12 10:23:31	a	2019-06-12 10:23:31
11	d	2019-06-12 10:23:31	a	2019-06-12 10:23:31
12	d	2019-06-12 10:23:31	a	2019-06-12 10:23:31
13	d	2019-06-12 10:23:31	a	2019-06-12 10:23:31
14	d	2019-06-12 10:23:31	a	2019-06-12 10:23:31

Figura 8.1: Vista de base de datos .

métodos ocupados en Java:

```

@Entity
@Table(name="Actuador1")
@EntityListeners(AuditingEntityListener.class)
@JsonIgnoreProperties(value= {"creationDate","creationTime"},allowGetters=true)
public class Note {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @NotBlank
    private String estado;

    @Column(nullable=false,updatable=false)
    @Temporal(TemporalType.DATE)
    @CreatedDate
    private Date creationDate;

    @Column(nullable=false,updatable=false)
    @Temporal(TemporalType.TIME)
    @CreatedDate
    private Date creationTime;

    @Column(nullable=false)
    @Temporal(TemporalType.TIMESTAMP)
    @LastModifiedDate
    private Date updateAt;
}

```

Figura 8.2: Código en Eclipse



## 9 Descripción de Prerrequisitos y Configuración

Para la realización del sistema se necesita una lista de programas y ya previamente instalados como:

- Docker
- Minikube
- kubernetes

### 9.0.1 Arduino IDE

Se debe tener instalada la última versión optimizada, por parte de la plataforma de Arduino, la cual posee compatibilidad, con la mayoría de librerías que se planea ocupar.

### 9.0.2 Eclipse

Se debe instalar también Eclipse la cual es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Esta plataforma es compatible directamente con SpringBoot y con Maven.

### 9.0.3 Librería NodeMCu

Se debe tener en cuenta la librería WiFi para ESP8266 ha sido desarrollada basándose en el SDK de ESP8266, usando nombres convencionales y la filosofía de funcionalidades generales de la librería WiFi de Arduino. Con el tiempo, la riqueza de las funciones WiFi del SDK de ESP8266 pasadas a ESP8266/Arduino superan a la librería WiFi de Arduino y se hizo evidente que tenemos que proporcionar documentación por separado

sobre lo que es nuevo y extra.

Esta documentación lo guiará a través de varias clases, métodos y propiedades de la librería ESP8266WiFi. Si eres nuevo en C++ y Arduino, no te preocupes. Comenzaremos por conceptos generales y luego pasaremos a la descripción detallada de los miembros de cada clase en particular, incluidos los ejemplos de uso.

El alcance de la funcionalidad que ofrece la biblioteca ESP8266WiFi es bastante extensa y por lo tanto, esta descripción se ha dividido en documentos separados marcados con :arrowright:.

---

# 10 Aportaciones

Utilizacion de Minikube para el despliegue de un clúster de Kubernetes con un único nodo en una máquina virtual.



# 11 Conclusiones

- El ESP8266 es muy versátil para el desarrollo de proyectos ya que es fácil de trabajar y acoplar sus módulos con lo cual nos da una amplia capacidad para realizar sistemas, aplicaciones en basadas en servicios IOT.
- Mediante la utilización de los servicios API-REST se puede tener mejores resultados, cuando se intentar controlar distintos actuadores, mediante servicios de internet, además de poder obtener en tiempo real valores de los sensores que se encuentren en un determinado entorno
- Con la facilidad que se realiza la conexiones con el ESP8266 se podría para un trabajo futuro realizar un sistema mas complejos donde se tendria mas dispositivos conectados de manera domótica.
- Las distintas conexiones con el nodeMCU se puede concluir que la ventaja de este módulo es poder incluir este tipo de conexiones en servicios de internet, y también encontrar dichas librerías para una óptima utilización de la programación realizada.





## 12 Recomendaciones

- Tener conocimientos básicos de lo que es el código arduino, Java y lo que representa los servicios API-REST.
- Comprender la naturaleza de los sensores y actuadores aplicados al sistema.
- Estudiar el diagrama de conexiones del arduino, y de los sensores ocupados.
- Se recomienda verificar detalladamente la estructura del prototipo a diseñar, teniendo ideas claras, y el enfoque a donde va dirigido.



# 13 Cronograma

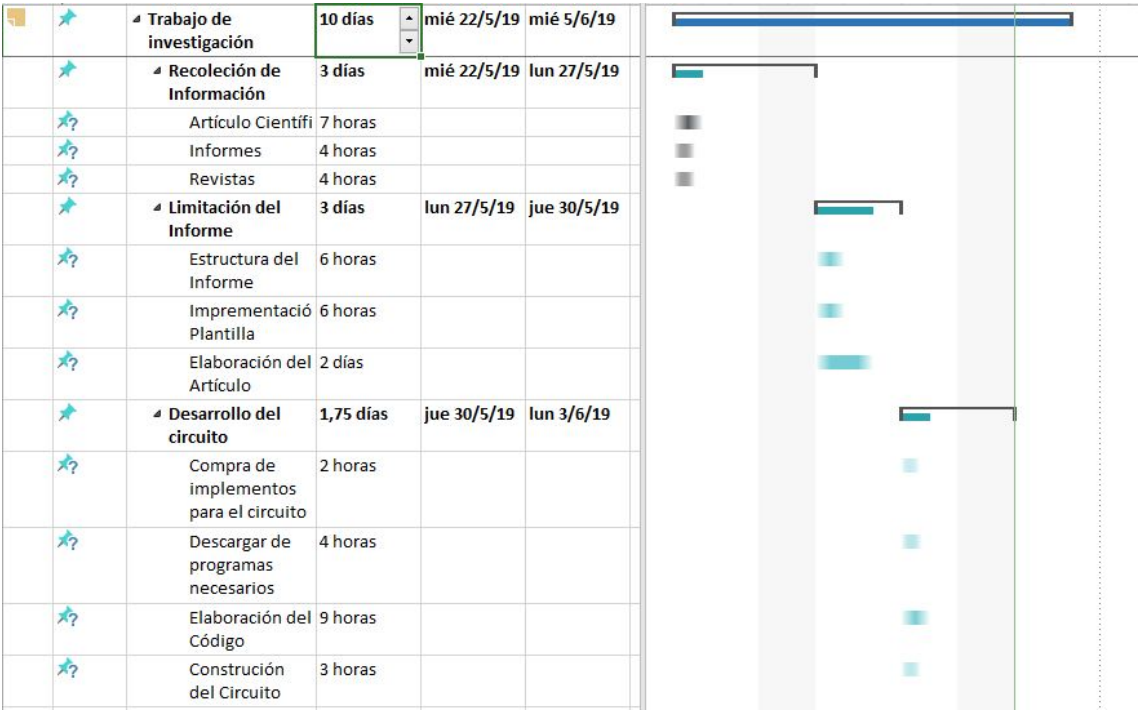


Figura 13.1: Cronograma de actividades parte 1



Figura 13.2: Cronograma de actividades parte 2



# 14 Repositorio

## 14.1 Git Hub

[https://github.com/Jhordyb3/Producto\\_unidad\\_2/blob/master/Producto\\_Unidad\\_2.zip](https://github.com/Jhordyb3/Producto_unidad_2/blob/master/Producto_Unidad_2.zip)

## 14.2 Links Overleaf

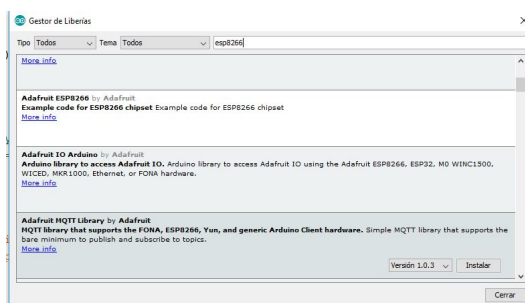
<https://es.overleaf.com/9464852251hdbcgcnjzsts>  
<https://es.overleaf.com/read/wmcypgtyftdc>



## 15 Anexos

## 15.1 Manual de usuario

Primeramente se debe tener instalada todas las librerías necesarias tanto para los sensores, como para el NODEMCU. Después se debe verificar los datos que serán



**Figura 15.1:** Librería ESP

enviados mediante el nodeMCU.

```

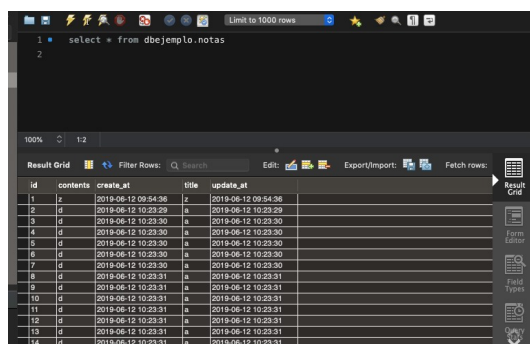
1 struct tnode {
2     int data; //data
3     struct tnode* lchild; //left child
4     struct tnode* rchild; //right child
5     struct tnode* p; //parent
6     struct tnode* q; //predecessor
7     struct tnode* s; //successor
8 }
9
10 struct tnode* createNode()
11 {
12     struct tnode* n;
13     n = (struct tnode*) malloc(sizeof(struct tnode));
14     if (n == NULL)
15         return NULL;
16     return n;
17 }
18
19 void insert(struct tnode* root, int data)
20 {
21     struct tnode* n;
22     n = createNode();
23     if (n == NULL)
24         return;
25     n->data = data;
26     if (root == NULL)
27         root = n;
28     else
29     {
30         struct tnode* p = root;
31         while (p->lchild != NULL || p->rchild != NULL)
32         {
33             if (p->lchild == NULL)
34             {
35                 p->lchild = n;
36                 return;
37             }
38             if (p->rchild == NULL)
39             {
40                 p->rchild = n;
41                 return;
42             }
43             if (data < p->data)
44                 p = p->lchild;
45             else
46                 p = p->rchild;
47         }
48     }
49 }
50
51 struct tnode* delete(struct tnode* root, int data)
52 {
53     struct tnode* n;
54     n = createNode();
55     if (n == NULL)
56         return NULL;
57     n->data = data;
58     if (root == NULL)
59         root = n;
60     else
61     {
62         struct tnode* p = root;
63         while (p->lchild != NULL || p->rchild != NULL)
64         {
65             if (p->lchild == NULL)
66             {
67                 p->lchild = n;
68                 return;
69             }
70             if (p->rchild == NULL)
71             {
72                 p->rchild = n;
73                 return;
74             }
75             if (data < p->data)
76                 p = p->lchild;
77             else
78                 p = p->rchild;
79         }
80     }
81 }
82
83 void print(struct tnode* root)
84 {
85     if (root == NULL)
86         return;
87     print(root->lchild);
88     printf("%d ", root->data);
89     print(root->rchild);
90 }
91
92 int main()
93 {
94     struct tnode* root = NULL;
95     int data;
96     while (1)
97     {
98         printf("Enter data: ");
99         scanf("%d", &data);
100         if (data == -1)
101             break;
102         insert(root, data);
103     }
104     printf("Binary tree after insertion: ");
105     print(root);
106     printf("\n");
107     while (1)
108     {
109         printf("Enter data to delete: ");
110         scanf("%d", &data);
111         if (data == -1)
112             break;
113         delete(root, data);
114     }
115     printf("Binary tree after deletion: ");
116     print(root);
117     printf("\n");
118 }

```

**Figura 15.2:** Librería ESP

Con ello, se verificara como se encuentra la base datos, y si se esta recibiendo los datos de los respectivos sensores.

Por ultimo se controlara los actuadores que poseamos.



The screenshot shows a database management interface. At the top, a SQL query is entered: `1 select * from dbjemplo.notas`. Below the query, a table of results is displayed. The table has five columns: `id`, `contents`, `create_at`, `title`, and `update_at`. The data is as follows:

id	contents	create_at	title	update_at
1	z	2019-06-12 09:54:36	z	2019-06-12 09:54:36
2	d	2019-06-12 10:23:29	a	2019-06-12 10:23:29
3	d	2019-06-12 10:23:30	a	2019-06-12 10:23:30
4	d	2019-06-12 10:23:30	a	2019-06-12 10:23:30
5	d	2019-06-12 10:23:30	a	2019-06-12 10:23:30
6	d	2019-06-12 10:23:30	a	2019-06-12 10:23:30
7	d	2019-06-12 10:23:30	a	2019-06-12 10:23:30
8	d	2019-06-12 10:23:31	a	2019-06-12 10:23:31
9	d	2019-06-12 10:23:31	a	2019-06-12 10:23:31
10	d	2019-06-12 10:23:31	a	2019-06-12 10:23:31
11	d	2019-06-12 10:23:31	a	2019-06-12 10:23:31
12	d	2019-06-12 10:23:31	a	2019-06-12 10:23:31
13	d	2019-06-12 10:23:31	a	2019-06-12 10:23:31
14	d	2019-06-12 10:23:31	a	2019-06-12 10:23:31

Figura 15.3: Librería ESP



# Bibliografía

- [1] Francisco Cilleruelo. Gestor de transacciones distribuidas asíncronas en arquitecturas de microservicios Máster Universitario de Investigación en Ingeniería de Software y Sistemas Informáticos Itinerario de Ingeniería de Software Gestor de transacciones distribuidas asíncronas. 2017.
- [2] D Barredo Gil. Escuela politécnica de ingeniería de gijón. 2019.
- [3] Matías Gabriel. Desarrollo e implementación de una arquitectura horizontalmente escalable. 2017.
- [4] Silvia Mariana and Mina García. Universidad técnica del norte universidad mariana de pasto instituto de postgrado. 2012.