

Monitoreo CO₂ en la Universidad Iberoamericana de Puebla.

1. Introducción

A partir de la pandemia COVID-19 las condiciones de vida cambiaron de manera súbita, a partir de este evento extraordinario la vida pública y todo lo que conlleva se vieron severamente afectados, obligando a la población mundial a acatar medidas sanitarias tanto personales como colectivas. La OMS (Organización Mundial de la Salud) reconoció la enfermedad como una pandemia global el día 11 de marzo de 2020; hoy, a poco más de un año de la fecha, la situación epidemiológica permite relajar de manera paulatina algunas de las medidas sanitarias impuestas a nivel nacional. Una de las actividades que fue golpeada con más fuerza debido a la pandemia fue el sector educativo en todos sus niveles, el regreso a las actividades diarias entonces comprende, por supuesto, el regreso paulatino y cuidadoso a las aulas del sector universitario.

La Universidad Iberoamericana de Puebla, entre otras universidades, ofreció la opción de regreso híbrido, abriendo parcialmente las aulas a sectores específicos de la población estudiantil; evidentemente, este regreso al mapus vino con condiciones, las más importantes el automonitoreo (que permite al alumno estar al pendiente de su propia salud y dar la oportunidad de detectar síntomas tempranos de COVID-19), uso obligatorio de cubrebocas y aforos específicos para cada espacio en el plantel.

Con el propósito de prestar ayuda a nuestra universidad este proyecto se centra en la elaboración de un semáforo a base de sensores de CO₂; esta propuesta permite realizar un cálculo más preciso del aforo para cada uno de los espacios donde sean colocados asegurando así un espacio sanitario y seguro para sus ocupantes

2. Marco teórico

HTML es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet. Se trata de las siglas que corresponden a HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto, esto no se debe confundir ya que HTML sólo sirve para indicar cómo va ordenado el contenido de una página web. Esto lo hace por medio de las marcas de hipertexto las cuales son etiquetas conocidas en inglés como tags.

CSS (en inglés Cascading Style Sheets) es lo que se denomina lenguaje de hojas de estilo en cascada y se usa para estilizar elementos escritos en un lenguaje de marcado como HTML. CSS separa el contenido de la representación visual del sitio.

La relación entre HTML y CSS es muy fuerte. Dado que HTML es un lenguaje de marcado (es decir, constituye la base de un sitio) y CSS enfatiza el estilo (toda la parte estética de un sitio web), van de la mano. CSS no es técnicamente una necesidad, pero muestra una mejor presentación a la página y no se queda en una forma tan estándar.

JavaScript es un lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web, cada vez que una página web hace algo más que sentarse allí y mostrar información estática para que la veas, muestra oportunas actualizaciones de contenido, mapas interactivos, animación de Gráficos 2D/3D, desplazamiento de máquinas reproductoras de vídeo, etc., puedes apostar que

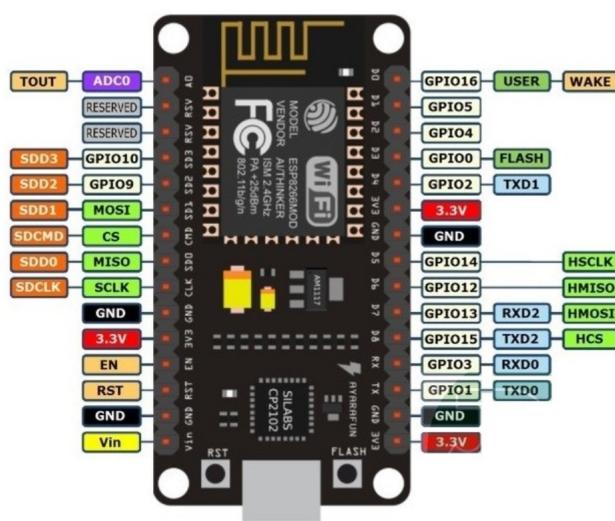
probablemente JavaScript está involucrado. Es la tercera capa del pastel de las tecnologías web estándar, dos de las cuales (HTML y CSS) hemos cubierto con mucho más detalle en otras partes del Área de aprendizaje.

El elemento HTML <input> se usa para crear controles interactivos para formularios basados en la web con el fin de recibir datos del usuario. Hay disponible una amplia variedad de tipos de datos de entrada y widgets de control, que dependen del dispositivo y el agente de usuario (user agent). El elemento <input> es uno de los más potentes y complejos en todo HTML debido a la gran cantidad de combinaciones de tipos y atributos de entrada.

Hypertext Preprocessor es un lenguaje de código abierto utilizado para desarrollo web con la capacidad de incrustación en HTML. Lo que distingue a PHP de algo del lado del cliente como JavaScript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga.

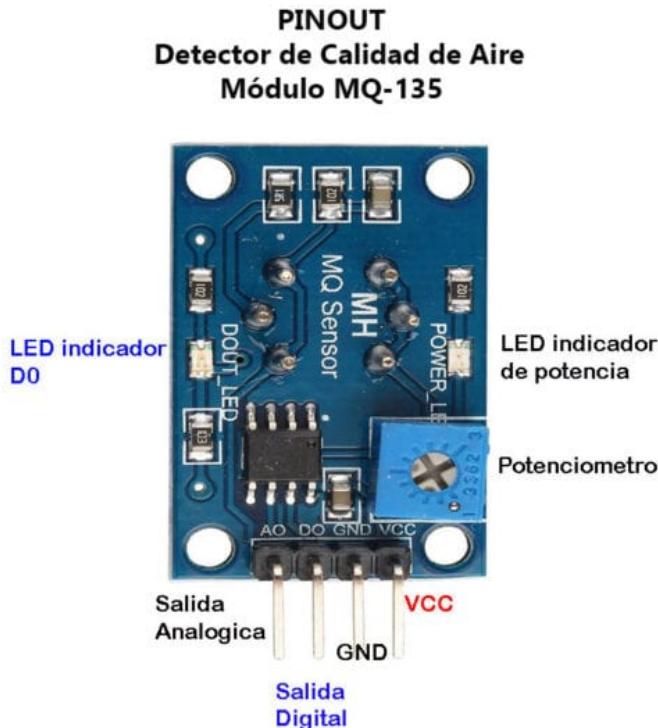
NodeMCU es una placa de desarrollo que integra SoC ESP8266, un chip de bajo costo Wi-Fi con un stack TCP/IP y un microcontrolador, este tipo de placas han evolucionado con los años, de la mano de distintos fabricantes.

NodeMCU es un nombre que recoge firmware de Open Source el cual es posible grabar en el chip Wi-Fi que permite programar con lenguaje script Lua, también es posible utilizar C++ con el entorno de Arduino y otras como Micro Python. La placa de desarrollo NodeMCU está basada en el ESP12E que permite funcionalidades y ventajas como programación sencilla a través de Micro-USB, terminales en pines que facilitan la conexión junto a un LED y botón de reset integrados.



Este sensor de control de calidad de aire es usado para la detección de contaminación en el medio ambiente, por lo general es implementado en circuitos de control como alarmas en las casas, sitios donde se desea prevenir altos niveles de contaminación a nivel aeróbico como industrias que manejan compuestos químicos que pueden ser nocivos también para la salud, especialmente en equipos controladores de calidad de aire en edificios/oficinas.

Este sensor se encarga de la detección de concentración de gas en diversos porcentajes, tal y como los son sus análogos MQ-3/4/5. La señal de salida que proporciona el MQ-135 es dual, de carácter analógico y digital. Respecto a la señal analógica proporcionada, esta viene a ser directamente proporcional al incremento de voltaje. La señal digital, esta presenta niveles TTL por lo que esta señal puede ser procesada por un microcontrolador.



3. Objetivos generales:

- Construcción de un sistema de monitoreo de CO₂ para salones en la Universidad Iberoamericana Puebla.

4. Objetivos específicos:

- Configurar el sensor de CO₂ correctamente para recibir lecturas precisas.
- Diseñar un programa capaz de detectar el nivel de CO₂ en nueve habitaciones para ser enviado a una página web haciendo uso del NodeMCU.
- En una página web diseñar un dashboard donde muestre la información mandada por el NodeMCU.

5. Desarrollo:

5.1 CSS

CSS está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación del mismo, en otras palabras creamos un estándar para las características de la página web como los colores y las fuentes. En la actualidad, existen innumerables plantillas CSS que facilitan la edición de páginas web.

```
@import url('https://fonts.googleapis.com/css2?family=Raleway:wght@300;400;700&display=swap');

.titulo{
    font-size:20px ;
    font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif ;
    color: white;
}

.tabla_titulo{
    background-color: #FF314D;
    height: 33px;
}

.tabla_sensor{
    background-color: white;
    height: 177px;
    border-color: #707070;
    border-width: 1px;
    border-style: solid;
    justify-content: center;
}

.tabla_inicio{
    background-color: white;
    height: 177px;
    justify-content: center;
}

.texto_grande2{
    font-size:21px ;
    font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif ;
    color: #707070;
    line-height: 1;
}

.texto_grande{
    font-size:20px ;
    font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif ;
    color: #707070;
    line-height: 0;
}
```

La manera de utilizarlo es mediante clases, para crearlas es necesario colocar un punto (".") al inicio de cada una, tal cual se aprecia en la imagen, y asignarle un nombre. Dentro de los corchetes se modifican los parámetros que van a modificar el html. Además de utilizar las clases predeterminadas se añadieron otras configuraciones al archivo CSS para integrar de forma correcta las imágenes utilizadas en la página web.

5.2 Header

```
<body>
  <header class="hero">

    <div class="container">
      <nav class="nav">
        <a href="index.html" class="nav_items nav_items--cta">INICIO</a>
        <a href="iniciosesion.html" class="nav_items nav_items--cta">INICIA SESIÓN</a>
        <a href="Contacto.html" class="nav_items nav_items--cta">CONTACTO</a>

      </nav>
      <section class="hero__container">
        | 
        | <div class="hero__texts">
        |   <h1 class="hero__title">UNIVERSIDAD IBEROAMERICANA PUEBLA</h1>
        |   <h1 class="hero__title">PROYECTO FINAL REDES DIGITALES</h1>

        |   <a href="index.html" class="hero__cta">INICIO</a>
      </div>
    </section>
  </div>
  <div class="hero__wave" style="overflow:hidden;"><svg viewBox="0 0 500 150" preserveAspectRatio="none" style="height: 100%; width: 100%"></svg></div>
</header>
```

El elemento de HTML Header (<header>) representa un grupo de ayudas introductorias o de navegación. Puede contener algunos elementos de encabezado, así como también un logo, un formulario de búsqueda, un nombre de autor y otros componentes. Para la página de inicio contamos con 3 divisiones:

- **Botones de inicio, inicio de sesión y contacto:** Utilizando la etiqueta <nav> se establecieron como elementos de navegación, se les dio color y forma con estilo CSS, y usando el atributo “href” se pegaron los enlaces correspondientes.
- **Logo, imagen de fondo y título:** Usando la etiqueta <div> encontramos el título del proyecto, la imagen de fondo y el logo, los cuales usando estilo CSS se les dio color, se centró el texto y se le puso un efecto a la imagen de fondo.
- **Línea límite:** Como última división tenemos la línea que limita el encabezado, la cual fue hecha a base de estilo CSS y con la etiqueta <path> para dibujar la curva.

5.3 Footer

```
<footer class="footer">
  <div class="container footer_grid">
    <nav class="nav nav--footer">
      <a class="nav_items nav_items--footer" href="">Inicio</a>
      <a class="nav_items nav_items--footer" href="">Acerca del proyecto</a>
      <a class="nav_items nav_items--footer" href="">Sobre los sensores</a>
      <a class="nav_items nav_items--footer" href="">Información extra</a>
      <a class="nav_items nav_items--footer" href="">Profesores encargados</a>
    </nav>

    <section class="footer_contact">
      
      <h3 class="footer_title">Contacto</h3>
      <div class="footer_icons">
        <span class="footer_container-icons">
          | <a class="fab fas fa-envelope footer_icon" href="https://outlook.office.com/mail/"></a>
        </span>

        <span class="footer_container-icons">
          | <a class="fab fa-whatsapp footer_icon" href="https://wa.me/2227543612"></a>
        </span>

        <span class="footer_container-icons">
          | <a class="fab fas fa-envelope footer_icon" href="https://mail.google.com"></a>
        </span>
      </div>
    </section>
  </div>
</footer>
```

El elemento HTML Footer (<footer>) representa un pie de página para el contenido de sección más cercano o el elemento raíz de sección. Un pie de página típicamente contiene

información acerca del autor de la sección, datos de derechos de autor o enlaces a documentos relacionados. El footer hecho para la página contiene elementos similares al header, en este caso contamos con dos divisiones:

- **Barra de navegación:** Usando la etiqueta `<nav>` se introdujeron cinco elementos en forma de tabla, en este caso se optó por un diseño más sencillo a comparación del header, de igual forma con estilo CSS.
- **Contacto:** Esta división fue hecha con la etiqueta `<section>` y luego dividida con la etiqueta `<div>`. Fuera del `<div>` encontramos la imagen del logo de la Universidad Iberoamericana Puebla y dentro, usando estilo CSS, encontramos las figuras para los contactos directos enlazadas con el atributo “`href`” a sus enlaces.

5.4 Creación de nuevo usuario

The screenshot shows a user creation form titled "Monitoreo CO2-IBERO". At the top left is the IBERO Puebla logo. To its right is another logo for "IBERO PUEBLA ®" and a small grey box labeled "Departamento de Ciencias e Ingenierías". Below the logo is a red header bar with the text "Monitoreo CO2-IBERO". The main form area has a white background. It starts with the heading "Nuevo Usuario:". Below it are four input fields: "Usuario:" (with a placeholder box), "Contraseña:" (with a placeholder box), "Verificar contraseña:" (with a placeholder box), and a "Crear nuevo usuario" button at the bottom.

IBERO
PUEBLA ®

IBERO
PUEBLA ®

Departamento de
Ciencias e
Ingenierías

Monitoreo CO2-IBERO

Ingresa tus datos para crear un nuevo usuario

Nuevo Usuario:

Usuario:

Contraseña:

Verificar contraseña:

Crear nuevo usuario

```
<table width=307px height=177px align=center class="tabla_inicio">
<tr >
    <td width=100%>
        <p></p>
        <p></p>
        <p class="texto_mediano" align=center>Ingresa tus datos para crear un nuevo usuario</p>
        <br>
        <br>
        <p align=center><b class="texto_grande2" >Nuevo Usuario:</b></p>
        <p></p>
        <b class="texto_grande2" >Usuario:</b>
        <p><input class="textbox" type="text" id="text_usuario"></p>
        <b class="texto_grande2" >Contraseña:</b>
        <p><input class="textbox" type="password" id="text_contra"></p>
        <b class="texto_grande2" >Verificar contraseña:</b>
        <p><input class="textbox" type="password" id="text_confir"></p>
        <p></p>
        <p align=center><input align=center class="boton" type="button" onclick="crear_nuevo_usuario()" value="Crear nuevo usuario"></p>
        <p></p>
    </td>
</tr>
</table>
```

Para la construcción de la página de “crear nuevo usuario” primero se utilizaron tanto el header y el footer que se explicaron anteriormente, y en el body tenemos la sección donde le pedimos al usuario de su nombre, cree su contraseña y la confirme para que posteriormente acceda con esos datos para revisar los sensores de CO2.

```
function crear_nuevo_usuario(){
    var usuario="";
    var contra="";
    var confir="";

    usuario=document.getElementById("text_usuario").value;
    contra=document.getElementById("text_contra").value;
    confir=document.getElementById("text_confir").value;

    //var cadena_texto=usuario+contra+confir;
    //alert(cadena_texto);

    if(usuario!="" && contra!="" && confir!=""){
        if(contra==confir){
            if(window.confirm("Nuevo usuario creado")){
                window.location.href="usuario.php?usuario="+usuario+"&contrasena="+contra;
            }
        } else{
            alert("Las contraseñas no coinciden");
        }
    } else{
        alert("Faltan campos por llenar");
    }
}

<?php

if(!file_exists("usuario.txt")){
    file_put_contents("usuario.txt", "");
}

if(isset($_GET["usuario"])&&isset($_GET["contrasena"])){
    $usuario=$_GET["usuario"];
    $contrasena=$_GET["contrasena"];
    $texto=$usuario."\r\n".$contrasena;
    file_put_contents("usuario.txt", $texto);
}

$archivo=file_get_contents("usuario.txt");
?>

<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="refresh" content="0; url=../iniciosesion.html">
    </head>
    <body>
    </body>
</html>
```

Teniendo ya el front end, ahora pasaremos al back end que está conformado por un archivo php y un archivo javascript. Como primera instancia tenemos el javascript, que se encarga de comparar los 3 campos: usuario, contraseña y confirmación, cada uno con su respectiva variable.

Desde el front end se toma id que se le designó a cada rubro y se guarda en una variable, paso siguiente usando lenguaje semejante al C, hacemos una comparación teniendo 3 posibles mensajes emergentes:

- Si alguno o algunos rubros no están completos, saltará el mensaje “Faltan campos por llenar”.
- Si la confirmación no es igual a la contraseña, saltará el mensaje “las contraseñas no coinciden”.
- Si los campos están completos y la confirmación es igual a la contraseña, entonces

saltará el mensaje “Nuevo usuario creado”.

Ahora pasando al php, una vez que se confirma la tercera opción, el javascript mandará la orden de que se ejecute el código de php que tiene como objetivo guardar la información dada por el usuario.

Primero el php realiza una comparación para verificar que si existe o no el archivo “.txt” donde se guardará la información de los usuarios. Por el contrario, el programa creará un archivo “.txt” recabando la información obtenida del javascript y dándole un formato para que posteriormente sea leído en la página de inicio de sesión.

5.5 Inicio de sesión

Bienvenido al sistema de monitoreo de CO2 de la universidad Iberoamericana Puebla

Inicia sesión

Usuario

Contraseña

¿Olvidaste la contraseña?, da clic [aqui](#)

Iniciar

Si aun no tienes usuario da clic [aqui](#)

```
<table width=307px height=177px align=center class="tabla_inicio">
<tr >
  <td width=100%>
    <p></p>
    <p></p>
    <p class="texto_mediano" align=center>Bienvenido al sistema de monitoreo de CO2 de la universidad Iberoamericana Puebla</p>
    <p align=center><b class="texto_grande2" >Inicia sesión</b></p>
    <p></p>
    <b class="texto_grande2" >Usuario</b>
    <p><input class="textbox" type="text" id="text_usuario"></p>
    <b class="texto_grande2" >Contraseña</b>
    <p><input class="textbox" type="password" id="text_contra"></p>
    <label class="texto_min2">¿Olvidaste la contraseña?, da clic </label><a href="#">aqui</a>
    <p></p>
    <p align=center><input align=center class="boton" type="button" value="Iniciar" onclick="iniciar_sesion()"></p>
    <p></p>
    <label class="texto_min2">Si aun no tienes usuario da clic </label><a href="#">nuevo_usuario">aqui</a>
  </td>
</table>
```

Para la página de inicio de sesión se siguió el mismo procedimiento para el front end que para la página de crear usuario, con la diferencia de que solo se piden 2 rubros que son nombre de usuario y contraseña.

```
function iniciar_sesion(){
    var usuario="";
    var contra="";

    usuario=document.getElementById("text_usuario").value;
    contra=document.getElementById("text_contra").value;

    //var cadena_texto=usuario+contra+confir;
    //alert(cadena_texto);

    if(usuario!="" && contra!=""){
        window.location.href="login.php?usuario="+usuario+"&contrasena="+contra;
    }
    else{
        alert("Faltan campos por llenar");
    }
}
```

```
?php
$direccion="../";

if(isset($_GET["usuario"])&&!isset($_GET["contrasena"])){
    $usuario=$_GET["usuario"];
    $contra=$_GET["contrasena"];

    $archivo=file_get_contents("nuevo_usuario/usuario.txt");
    $pos=strpos($archivo, "\r\n");
    $usuario_archivo=substr($archivo,0,$pos);
    $contra_archivo=substr($archivo, $pos+2);

    if($usuario==$usuario_archivo && $contra==$contra_archivo){
        $direccion="../sensores_2.php";
    }

    echo $usuario;
    echo $contra;
    echo $usuario_archivo;
    echo $contra_archivo;
}

?>

<!DOCTYPE html>
<html>
    <head>

    </head>

    <body>
        <?php echo $direccion ?>
        <script>window.location.href="php echo $direccion?&gt;"&lt;/script&gt;
    &lt;/body&gt;
&lt;/html&gt;</pre

```

Para el back end, el código del javascript será más sencillo ya que solo se encargará de dar un aviso en el caso de que no se haya completado el formulario o bien dar la orden de ejecutar el php.

En esta página el php tiene un mayor peso, ya que este se encargará de dar acceso a la página de los sensores si es que coincide el nombre del usuario y contraseña que tiene en su base de datos. Como podemos ver en el código se usará primero un if para verificar los datos dados por el javascript, el programa revisará el archivo txt generado por la página de crear usuario y comparará esa información con la que se dio. Si esta es correcta, se redirecciónará a la página de los sensores.

5.6 Página de sensores de CO₂

The screenshot displays a web-based monitoring system for CO₂ levels. At the top, the logo of the Universidad Iberoamericana de Puebla is visible, along with navigation links for 'INICIO', 'INICIA SESIÓN', and 'CONTACTO'. The main title 'RESULTADOS MONITOREO DE NIVELES DE CO₂' is centered above a grid of nine sensor status indicators.

Sensor 1 J105
nan
Última actualización: 23/06/21 11:00am

Sano: 350 a 450
Viciado: 600-800
Peligro: >1000-1500
Peligro grave: >6000

Sensor 3 J102
Última actualización: 23/06/21 11:00am

Sano: 350 a 450
Viciado: 600-800
Peligro: >1000-1500
Peligro grave: >6000

Sensor 7 J105
Última actualización: 23/06/21 11:00am

Sano: 350 a 450
Viciado: 600-800
Peligro: >1000-1500
Peligro grave: >6000

Sensor 2 J104
5.000
Última actualización: 23/06/21 11:00am

Sano: 350 a 450
Viciado: 600-800
Peligro: >1000-1500
Peligro grave: >6000

Sensor 4 J105
Última actualización: 23/06/21 11:00am

Sano: 350 a 450
Viciado: 600-800
Peligro: >1000-1500
Peligro grave: >6000

Sensor 8 J104
Última actualización: 23/06/21 11:00am

Sano: 350 a 450
Viciado: 600-800
Peligro: >1000-1500
Peligro grave: >6000

Sensor 5 J104
Última actualización: 23/06/21 11:00am

Sano: 350 a 450
Viciado: 600-800
Peligro: >1000-1500
Peligro grave: >6000

Sensor 9 J102
Última actualización: 23/06/21 11:00am

Sano: 350 a 450
Viciado: 600-800
Peligro: >1000-1500
Peligro grave: >6000

Sensor 6 J102
Última actualización: 23/06/21 11:00am

Sano: 350 a 450
Viciado: 600-800
Peligro: >1000-1500
Peligro grave: >6000

```
<table width=400px height=180px align=center class="tabla_sensor">
<tr>
    <td align=center width=50%>
        <p></p>
        <p></p>
        
        <p class="texto_min">Sano: 0 a 100</p>
        <p class="texto_min">Viciado: 100-200</p>
        <p class="texto_min">Peligro: >100</p>
    </td>

    <td align=center width=50%>
        <p class="texto_grande"> Sensor 1</p>
        <b class="texto_grande"> J105</b>
        <p class="texto_grande"> <?php echo $s1 ?> </p>
        <p class="texto_min">Ultima actualización:</p>
        <p class="texto_min">23/06/21 11:00am</p>
    </td>
</tr>
</table>
```

```
<?php
    $s1=file_get_contents("sensor_1/datos_s1.txt");
    $s2=file_get_contents("sensor_2/datos_s2.txt");
    $s3=file_get_contents("sensor_3/datos_s3.txt");
    $s4=file_get_contents("sensor_4/datos_s4.txt");
    $s5=file_get_contents("sensor_5/datos_s5.txt");
    $s6=file_get_contents("sensor_6/datos_s6.txt");
    $s7=file_get_contents("sensor_7/datos_s7.txt");
    $s8=file_get_contents("sensor_8/datos_s8.txt");
    $s9=file_get_contents("sensor_9/datos_s9.txt");
?>
```

de cada sensor.

```
<?php
    if(!file_exists("datos_s1.txt")){
        file_put_contents("datos_s1.txt", "");
    }

    if( isset($_GET['A']) ){ //esta linea es para comprobar si ha recibido algo en alguno de los sensores
        $dato=$_GET['A'];
        file_put_contents("datos_s1.txt", $dato);
    }

    $ARCHIVO=file_get_contents("datos_s1.txt"); //consigue del archivo txt, ARCHIVO es todo el texto del documento
?>

<!DOCTYPE html>
<html>
    <head>
        <title>Sensores PHP</title>
        <meta http-equiv="refresh" content="5"> <!-- pag se actualiza cada 5 segundos -->
    </head>
    <body>
        <p> <?php echo $ARCHIVO; ?> </p>
    </body>
</html>
```

Para recabar la información obtenida para cada NodeMCU de los sensores MQ135, se creó un archivo php para cada dispositivo, donde se recibirá la lectura (ya convertida a ppm) con un request de tipo http por parte del NodeMCU y se guardará en un archivo ".txt". El archivo se irá actualizando cada cierto periodo dependiendo del tiempo establecido en el envío de

Para la página de los sensores de CO2 contamos con múltiples partes que la conforman, la primera sería el front end donde se sigue la misma estética que las anteriores páginas, utilizando el header, el footer y para este caso, se realizaron múltiples tablas donde se encontrará la información de cada uno de los sensores.

Como podemos ver en la imagen anterior la tabla conforma los datos generales del estado en el que se encuentra el cuarto donde se ubica el sensor. La línea más importante es la que tiene como clase = "texto_grande" donde hace referencia al valor de la variable \$s1.

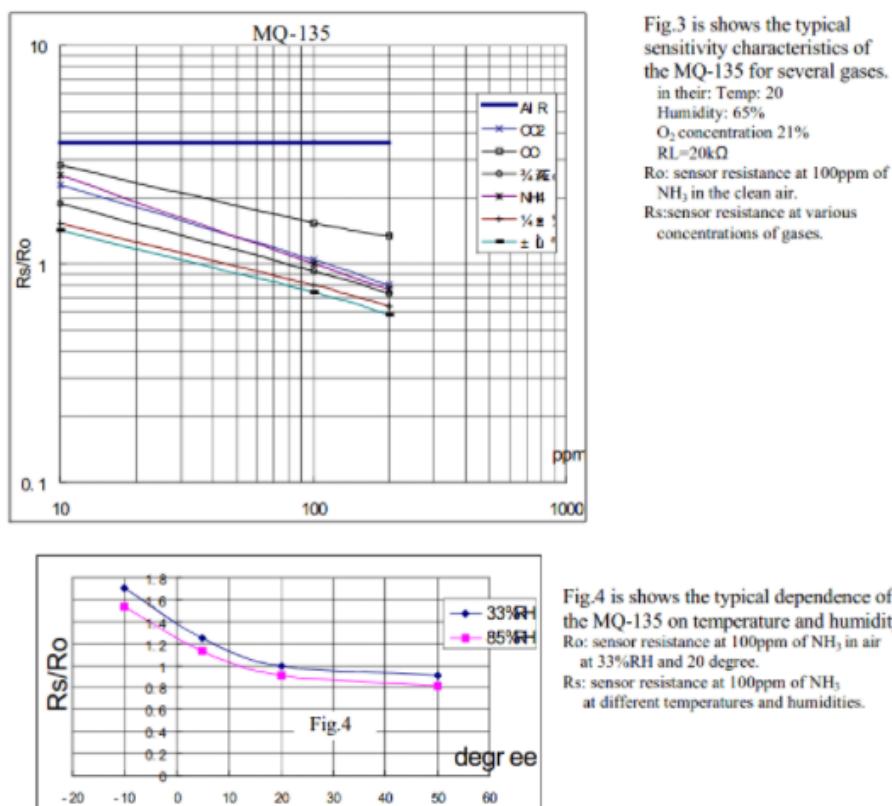
La variable \$s1 y en adelante están referenciadas en la parte superior del código donde encontramos un php que guarda en esas variables los datos contenidos en los archivos ".txt" dentro de la carpeta

datos de los dispositivos y eso se verá reflejado en las tablas del dashboard de la página web.

5.7 Configuración de los sensores

Para calibrar los sensores es necesario activarlos para medir aire “limpio”, de acuerdo con las estadísticas, se sabe que en la actualidad las partículas por millón de CO₂ en aire limpio son de 418.31 ppm (parts per million, por sus siglas en inglés). El sensor mide la cantidad de CO₂ con respecto a una resistencia interna varía de acuerdo a la cantidad de gas presente. Para poder calibrarlos es necesario medir aire limpio durante media hora, para esto se usarán una serie de cálculos que serán explicados a continuación. Primero se hablará de los términos que conforman las ecuaciones necesarias y luego los cálculos realizados a nivel programación.

Revisando la datasheet del sensor utilizado, podemos encontrar las gráficas en la imagen siguiente, que muestran las características de sensibilidad del MQ-135, de inmediato se nota como no solo son difíciles de leer y entender, sino que al ser logarítmicas no pueden ser usadas de manera directa en la plataforma arduino, lo que hacen los cálculos realizados es precisamente eso, ajustar la curva.



Las ecuaciones que se explicarán son las siguientes:

$$R_0 = \frac{R_s}{a * ppm^b} \quad (1)$$

$$R_s = 1024\left(\frac{R_L}{adc}\right) - R_L \quad (2)$$

$$ppm = \left(\frac{\frac{Rs}{R_0}}{a} \right)^{1/b} \quad (3)$$

$$a = 5.5973021420 \quad (4)$$

$$b = -0.365425824 \quad (5)$$

Antes de explicar cada término de manera individual, vale la pena mencionar que los valores constantes a y b fueron obtenidos de la gráfica.

- **R₀**: Medida del sensor a 100ppm, este valor es constante y particular para cada uno de los sensores utilizados, esta diferencia se debe a las ligeras diferencias existentes en las resistencias que forman parte del sensor; en palabras más sencillas, nueve sensores, nueve valores distintos.
- **R_s**: Media de los valores medidos por Arduino cada segundo durante 5 minutos y la resistencia del sensor, se usa para la calibración del sensor.
- **R_L**: Carga de la resistencia, en este caso es de 20kΩ para todos los sensores.
- **adc**: Valor detectado por el sensor y leído por Arduino durante cinco minutos.
- **ppm**: Número de unidades de masa de contaminantes por millón de unidades de masa total. Se usa para medir la concentración de contaminantes en un medio.

Habiendo comprendido esto, se puede proceder a explicar los códigos usados en el proyecto. En orden, el cálculo de la variable R₀ (se recalca, como el valor será única para cada uno de los nueve sensores utilizados), el cálculo de ppm es decir el nivel de saturación para el CO₂ y finalmente el programa que se encarga de conectarse con internet y enviar los valores obtenidos.

5.7.1 Cálculo de R₀

```

float adc = 0, rs = 0, ro = 0;

void setup() {
    // put your setup code here, to run once:

    Serial.begin(115200);
    pinMode(0, INPUT);
}

void loop() {
    // put your main code here, to run repeatedly:

    for (int i=0; i<300; i++)
    {
        adc = analogRead(0);
        rs = 1024 * (20000/adc) - 20000;
        ro = rs/(5.597302142*(pow(418.31,-0.365425824)));
        Serial.println(ro);
        delay(1000);
    }

    Serial.println(".....");
    Serial.println("Proceso terminado");
    delay(30000);
}

```

En primer lugar se establecen las variables necesarias, del tipo float e inicializadas con un valor de cero. También se establecen los baudios necesarios, así como la modalidad del pin utilizado. En la sección `void loop()` del código se realizan 300 repeticiones con un delay de un segundo que se encarga del cálculo de la variable R_o , se recalca que al ser único para los sensores este valor el valor presentado en esta explicación es meramente ilustrativo; usando la ecuación correspondiente para R_s y el valor de adc que es un valor obtenido por el sensor y capturado por el Arduino UNO. De la línea que contiene la ecuación propia de R_o vale la pena mencionar que la instrucción `pow` sigue la sintaxis `pow(base, exponente)`.

Debido a las 300 repeticiones realizadas en el bucle, las cuales todas fueron un cálculo individual de R_o , se hizo uso de excel para poder sacar un promedio de todas ellas.

5.7.2 Envío de datos

```

//Sensor 2 Sin marca con un valor de Ro = 84299

#include <ESP8266WiFi.h> // Conexion del NodeMCU a la red Wifi Local
#include <WiFiClient.h> // Configurar NodeMCU cliente
#include <ESP8266HTTPClient.h> //Funciones enviar request POST o GET Cliente

//red Wifi
const char* ssid = "IBERO"; // Nombre de la red Wifi
const char* pass = ""; //Contraseña red Wifi

```

Se especifican las librerías necesarias; como lo muestra la imagen gracias a ellas se puede conectar a WiFi y configurar la modalidad del NodeMCU, así como enviar requests. Para conectarse a una red es necesario especificar su nombre y contraseña, en este caso se

seleccionó la red escolar, IBERO, que al ser pública carece de contraseña y por lo tanto solo requiere las obligatorias comillas.

```
//Pagina PHP
String serverName = "http://manu-redes.000webhostapp.com/sensores/sensor_1";
String v1 = "A";
String message="";

int adc=0;
float ppm=0;

float sensor();
```

Se especifican las instrucciones de envío a una página específica, así como los nombres que asocian los datos enviados a dicha página. Es importante mencionar que la URL mostrada es la que tiene como objetivo mandar los valores al sensor 1, al haber nueve sensores en total, hay nueve páginas con básicamente la misma URL, excepto que con el sensor correspondiente. Se inicializan las variables mostradas.

```
void setup() {
    // put your setup code here, to run once:

    Serial.begin(115200); //Inicializamos el serial
    delay(10); // Retraso esperamos se initialize el Serial

    conectarWifi(); //Funcion Local paraa conectarnos a Wifi
}
```

Se especifican los baudios necesarios para el funcionamiento correcto del NodeMCU, el delay está presente para permitir la iniciación correcta del serial. Se llama la función que permite la conexión a Internet, se explica abajo.

```
void loop() {
    // put your main code here, to run repeatedly:

    //sensor();
    -----
    float datol;

    datol= sensor();

    enviardato(datol); //Funcion Local para enviar datos a Thingspeak

    delay(1000); //Esperamos 5 segundos
}
```

Se ejecuta la función `sensor()` que se iguala al dato que será enviado, para luego ser mandado por la función `enviardato()`. El delay es una medida precautoria que evita la saturación de la página.

```
float sensor(){
    float Rs,Ro,a,b;
    int RL = 20000;

    adc = analogRead(0);
    a = 5.59730214;
    b = -0.36542582;
    Ro = 84299;

    Serial.println(adc);

    ppm = ((1024 * (RL/adc) - 20000) / Ro) / a;

    if(ppm<0)
        ppm = -ppm;

    Serial.println(ppm);
    Serial.println("ppm antes de potencia");

    ppm = pow(ppm, (1/b));
    Serial.println(ppm);
    Serial.println("ppm despues de potencia");

    return ppm;
}
```

Usando los valores definidos en el código anterior se definen los valores que serán usados en el cálculo del valor correspondiente a *ppm*. El valor usado para la variable *Ro* es el promedio de las 300 iteraciones realizadas dentro del bucle del código explicado arriba. Esta porción del código es la encargada del cálculo de *ppm*. Primero se muestra la fórmula, pero si nos remontamos a la fórmula 3 arriba es notable la ausencia del exponente $\frac{1}{b}$ esto es porque se deben hacer consideraciones especiales antes de poder aplicarlo. La consideración se ve reflejada en el *if* para asegurar que el cálculo siempre sea posible, se da la instrucción de que en caso de que *ppm* sea menor a cero, es decir un número negativo, se haga el cambio de signo a uno positivo. Asegurando lo anterior ya se aplica la potencia, obteniendo el valor que le corresponde a *ppm*. El valor obtenido será el dato enviado a la página correspondiente según su numeración.

```

void conectarWifi() {
    Serial.println(""); //Mostramos un espacio enter
    Serial.print("Connecting to "); //Print Conectando..
    Serial.println(ssid); //Muestra la red a la que se conecta
    WiFi.begin(ssid, pass); //Funcion que conecta la red Wifi

    while (WiFi.status() != WL_CONNECTED) //Ciclo de espera hasta que se conecte a red Wifi
    {
        delay(500); //esperamos 0.5 segundos
        Serial.print(".");
    }
    Serial.println(""); //Mostramos un espacio enter
    Serial.println("WiFi connected"); //Mostramos que ya se conecto
}

```

Esta función permite la conexión la red especificada al principio por código al NodeMCU; usando la consola el código permite comprobar que la conexión fue exitosa.

```

void enviardato(float dato1) {

    if(WiFi.status()== WL_CONNECTED) //Si el Wifi esta conectado?
    {
        HTTPClient http; //Creando un objeto de tipo HTTP Client
        message=serverName+"?"+vl+"="+String(dato1,3);
        http.begin(message.c_str()); //funcion para crear el request http

        Serial.print("HTTP Request Data: "); //Mostramos el texto
        Serial.println(message); //Mostramos el valor de Request Data

        int httpResponseCode=http.GET();
        //Funcion GET, realiza un request tipo GET con la información de la
        //Finalmente nos regresa el valor de respuesta del servidor, idealmente "200"

        Serial.println(dato1);
        Serial.print("HTTP Response code: "); //Mostramos el texto
        Serial.println(httpResponseCode); //Mostramos el valor de la respuesta
        Serial.println(""); //Mostramos un espacio enter

        http.end(); //Cerramos la conexion con el servidor
    }
    else //Si el Wifi esta desconectado?
    {
        Serial.println("WiFi Disconnected"); //Mostramos que esta desconectado
    }
}

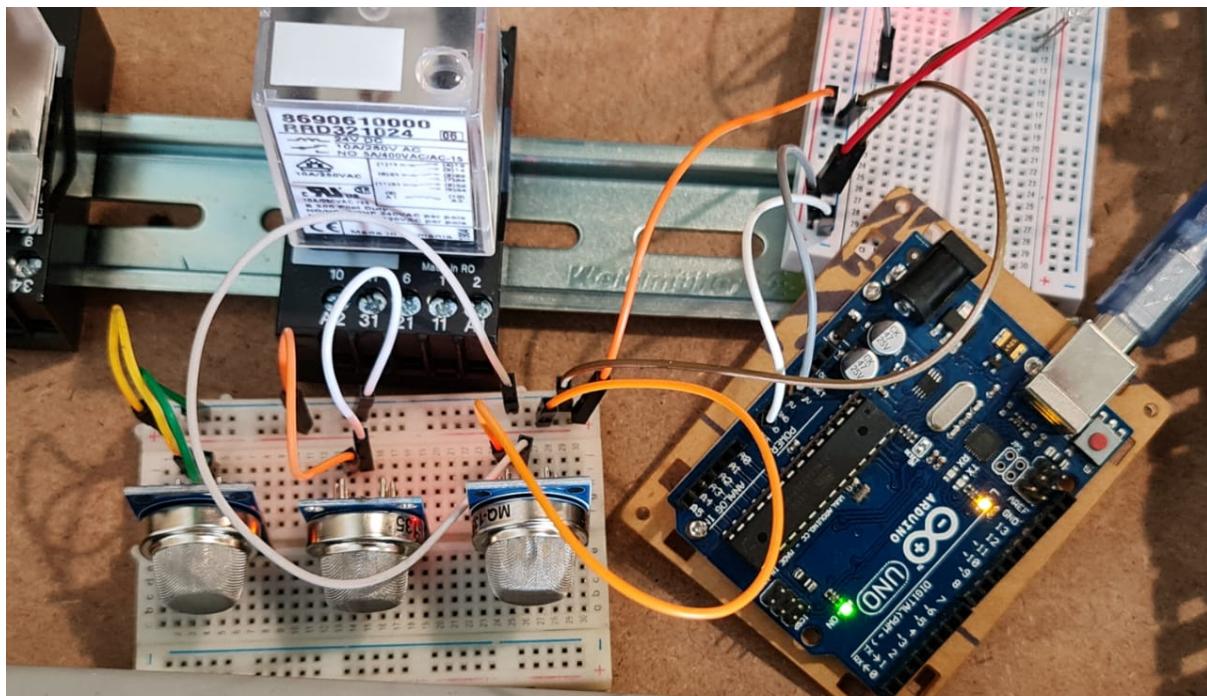
```

De ser exitosa la conexión entra en función la creación de un documento tipo *http* que es la dirección a donde se manda la información; *message* es la información mandada por *http* esta línea es muy importante puesto que especifica la estructura necesaria para el request, esto se nota por la presencia del signo de interrogación, el nombre asignado al sensor, el símbolo de igual y el *string* que convierte el *dato1* de float a texto (el número tres indica el tipo de conversión); la instrucción *http.begin()* crea el request que indica que *message* de tipo *string* se envía desde el objeto *http*. Como confirmación se imprime en la consola de Arduino el mensaje enviado. La instrucción *httpResponseCode* permite devolver el estado

del servidor, de ser exitoso el envío y la recepción del mensaje, se imprimirá en la consola el número 200 (este es dado por convención) de otro modo muestra diferentes valores. Al haber recibido el mensaje de manera exitosa se cierra la conexión, hasta nuevo aviso. Este set de instrucciones se conecta y desconecta cada ciclo, los cuales tienen una duración de aprox. un segundo entre envío y envío.

Si la conexión a internet no fue posible se manda alerta al usuario.

5.7.3 Conexión física



El modelo de los sensores utilizados para medir el CO₂ es el MQ135. Estos sensores son electroquímicos y varían su resistencia cuando se exponen a determinados gases. Internamente poseen un calentador encargado de aumentar la temperatura interna y con esto el sensor pueda reaccionar con los gases provocando un cambio en el valor de la resistencia. Debido a esta pieza es necesario esperar un tiempo de calentamiento para que la salida sea estable y tenga las características que el fabricante muestra en sus datasheet, dicho tiempo dependiendo del modelo puede ser entre 12 y 48 horas. Los sensores se dejaron conectados durante 24 horas. La imagen superior muestra la conexión física de tres de los nueve sensores utilizados

6. Resultados

Como se vio en el apartado de desarrollo, se hicieron uso de todas las herramientas aprendidas a lo largo del semestre. Juntando los aprendizajes de la clase se construyó una página dedicada al monitoreo de CO₂ en la Universidad Iberoamericana Puebla, alojada en la URL <https://monitoreoco2.000webhostapp.com/index.html>. La página combina funcionalidad, contando con las funciones de creación de usuario, inicio de sesión y el mismo semáforo que era el eje central del proyecto, y la estética, haciendo uso de las técnicas aprendidas para la aplicación del estilo CSS.

7. Conclusión:

En conclusión, el proyecto integrador presentado fue un éxito. La construcción del semáforo es una buena forma de poner en práctica lo aprendido a lo largo del semestre, pero también es un ejemplo de la importancia de la ingeniería en situaciones extraordinarias como la que se está viviendo en medio de la actual pandemia, este proyecto presenta la propuesta de una alternativa que permite obtener cálculos más precisos en el aforo de los salones lo cual significa condiciones más seguras para la comunidad escolar, a pesar de que el alcance del proyecto actualmente está limitado, tiene altas expectativas para implementarse en un futuro cercano como medida sanitaria dentro de la universidad, misma que cuenta con los recursos y tiene como obligación asegurar la seguridad colectiva e incluso mantener esta propuesta una vez que la pandemia haya pasado, como medida preventiva y recordatorio de que la precaución sanitaria nunca está demás.

8. Referencias:

- [1] “Qué es HTML,” *CódigoFacilito*, 2021. <https://codigofacilito.com/articulos/que-es-html> (accessed Jul. 04, 2021).
- [2] “Sensor MQ-135 gas calidad de aire,” *Naylamp Mechatronics - Perú*, 2021. <https://naylampmechatronics.com/sensores-gas/73-sensor-mq-135-gas-calidad-aire.html> (accessed Jul. 04, 2021).
- [3] G. B, “¿Qué es CSS?,” *Tutoriales Hostinger*, Jan. 24, 2019. <https://www.hostinger.mx/tutoriales/que-es-css> (accessed Jul. 04, 2021).
- [4] “¿Qué es JavaScript? - Aprende sobre desarrollo web | MDN,” *Mozilla.org*, Jul. 04, 2021. https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript (accessed Jul. 04, 2021).
- [5] E. González, “¿Qué es PHP? y ¿Para qué sirve? Un potente lenguaje de programación para crear páginas web. (CU00803B),” *Aprender programar.com*, 2021. https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=492:que-es-php-y-ipara-que-sirve-un-potente-lenguaje-de-programacion-para-crear-paginas-web-cu00803b&catid=70&Itemid=193 (accessed Jul. 04, 2021).
- [6] “input - HTML: Lenguaje de etiquetas de hipertexto | MDN,” *Mozilla.org*, Jul. 02, 2021. <https://developer.mozilla.org/es/docs/Web/HTML/Element/input> (accessed Jul. 04, 2021).
- [7] <http://facebook.com/rufianenlared>, “MQ-135, CO₂ y la ventilación en tiempos de pandemia,” *Rufián en la Red*, Oct. 09, 2020. <https://rufianenlared.com/mq-135/> (accessed Jul. 04, 2021).
- [8] “Earth’s CO₂ Home Page,” *CO₂.Earth*, 2012. <https://www.co2.earth/> (accessed Jul. 04, 2021).