**TECNOLÓGICO NACIONAL DE MÉXICO**

**INSTITUTO TECNOLÓGICO DE TLÁHUAC**

**Manual de Prácticas para la asignatura de Sistemas Programables SCC-1023**

**SATCA: 2-2-4.**

**Carrera: Ingeniería en Sistemas Computacionales.**

**Alumnos:**

**Jiménez Tinoco Sergio Quetzal**

**Rios Gomez Karina Rubi**

**Docentes:**

**Claudia Amezcua Fierros.**

**Martín Ramón Cordero Ocampo.**



Tláhuac, CDMX 12-Diciembre-2024

|  |
| --- |
| **PRACTICA NO. 1** |
| **NOMBRE:** **Principios de una página web** |
| **COMPETENCIA:** Crea una página web donde puedas visualiza datos de un sensor. |
| **PROCEDIMIENTO:**  Para la primera practica primero elaboraremos una página web donde visualizaremos más adelante datos.   * <!DOCTYPE html>: Define el tipo de documento HTML5. * <html>: Es el contenedor principal de todo el código. * <head>: Contiene metadatos y el título de la página. * <body>: Contiene el contenido visible de la página. * <h1>: Define un encabezado importante. * <p>: Define un párrafo de texto. * <a>: Crea un enlace a otra página web.   <!DOCTYPE html>  <html lang="es">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Mi Primera Página HTML</title>  </head>  <body>  <h1>Hola, Mundo!</h1>  <p>Este es un párrafo de ejemplo en mi primera página web.</p>  </body>  </html>    ahora solo agrega la temperatura que desees    EJEMPLO: <p>La temperatura es de 22°C.</p>    Ahora conectaremos el ESP 32 a la página.  Como nuestro ESP32 va a actuar de servidor tenemos que configurar la dirección, la contraseña y que el ESP registre los datos del sensor.  Con el software de Arduino configuraremos toda la parte de los sensores y la parte de la página.  #include <WiFi.h>  #include <DHT.h>  const char\* ssid = **~~"tu\_SSID";~~** <-**Reemplaza con tu nombre de red WiFi**  const char\* password = **~~"tu\_PASSWORD";~~** <- **Reemplaza con tu contraseña WiFi**  **Define el pin al que está conectado el sensor DHT**  #define DHTPIN 4  #define DHTTYPE DHT22 **Usa DHT11 si estás usando ese sensor**  DHT dht(DHTPIN, DHTTYPE);  WiFiServer server(80);  void setup() {  Serial.begin(115200);  dht.begin();  **Conectarse a Wi-Fi**  WiFi.begin(ssid, password);  while (WiFi.status() != WL\_CONNECTED) {  delay(1000);  Serial.println("Conectando a WiFi...");  }  Serial.println("Conectado a WiFi");  **Iniciar el servidor**  server.begin();  }  void loop() {  WiFiClient client = server.available(); // Espera a un cliente  if (client) {  Serial.println("Nuevo cliente conectado");  String currentLine = "";    **Leer la temperatura**  float t = dht.readTemperature();    **Si la lectura falla, no enviamos datos**  if (isnan(t)) {  Serial.println("Error al leer el sensor DHT");  return;  }    **Responder al cliente con los datos HTML**  client.println("HTTP/1.1 200 OK");  client.println("Content-Type: text/html");  client.println("Connection: close");  client.println();    **Página web simple con la temperatura**  client.println("<!DOCTYPE html>");  client.println("<html lang='es'>");  client.println("<head><meta charset='UTF-8'><title>Temperatura ESP32</title></head>");  client.println("<body><h1>Temperatura Actual</h1>");  client.print("<p>La temperatura es de ");  client.print(t);  client.println(" °C.</p>");  client.println("</body>");  client.println("</html>");    **Espera antes de cerrar la conexión**  delay(1000);  }  }  **Como acceder a la página.**  Para acceder a la página tendremos que estar en la misma red Wi-Fi que se colocó al inicio.  const char\* ssid = **~~"tu\_SSID";~~** <-**Reemplaza con tu nombre de red WiFi**  const char\* password = **~~"tu\_PASSWORD";~~** <- **Reemplaza con tu contraseña WiFi**  El ESP32 devuelve una página HTML con la temperatura actual en el formato siguiente: |

|  |
| --- |
| **PRACTICA NO. 2** |
| **NOMBRE: Enciente y apaga un sensor** |
| **COMPETENCIA: Aprender el uso de sensores para la extracción de datos.** |
| **PROCEDIMIENTO:**  Con el anterior código donde programamos el ESP y el sensor vamos agregar el siguiente sensor que será de movimiento el cual vamos a poder prender y apagar cuando lo decimos.    Después de la configuración del primer sensor pondremos el segundo y lo configuraremos.    **Leer la temperatura**  float t = dht.readTemperature();    **Si la lectura falla, no enviamos datos**  if (isnan(t)) {  Serial.println("Error al leer el sensor DHT");  return;  }    **Leer el estado del sensor PIR**  bool motionDetected = digitalRead(PIRPIN);    **Procesar las peticiones de la web**  while (client.connected()) {  if (client.available()) {  char c = client.read();  currentLine += c;  **Si encontramos la petición de encender el sensor PIR**  if (currentLine.endsWith("GET /encender")) {  pirState = true;  }  **Si encontramos la petición de apagar el sensor PIR**  if (currentLine.endsWith("GET /apagar")) {  pirState = false;  }  **Si encontramos una línea en blanco (fin de la cabecera)**  if (currentLine.length() == 0) {  **Enviar la respuesta HTTP**  client.println("HTTP/1.1 200 OK");  client.println("Content-Type: text/html");  client.println("Connection: close");  client.println();    **Página web con la temperatura y el estado del sensor PIR**    client.println("<!DOCTYPE html>");  client.println("<html lang='es'>");  client.println("<head><meta charset='UTF-8'><title>Temperatura y Movimiento</title></head>");  client.println("<body><h1>Temperatura Actual</h1>");  client.print("<p>La temperatura es de ");  client.print(t);  client.println(" °C.</p>");    **Mostrar el estado del sensor PIR**  if (pirState) {  client.println("<p>El sensor PIR está activado.</p>");  } else {  client.println("<p>El sensor PIR está desactivado.</p>");  }  **Botones para encender o apagar el sensor PIR**  client.println("<a href='/encender'><button>Encender Sensor PIR</button></a>");  client.println("<a href='/apagar'><button>Apagar Sensor PIR</button></a>");  **Mostrar si se detectó movimiento**  if (motionDetected && pirState) {  client.println("<p><strong>¡Movimiento detectado!</strong></p>");  } else {  client.println("<p>No se ha detectado movimiento.</p>");  }  client.println("</body>");  client.println("</html>");    break;  }  }  }  client.stop();  **Cerrar la conexión con el cliente**  }  }  **Como ingresamos a la página.**  Para acceder a la página tendremos que estar en la misma red wi-fi que se colocó al inicio.  const char\* ssid = **~~"tu\_SSID";~~** <-**Reemplaza con tu nombre de red WiFi**  const char\* password = **~~"tu\_PASSWORD";~~** <- **Reemplaza con tu contraseña WiFi**  Así es como se ve cuando el sensor se encuentra apagado.  Para cuando el sensor este encendido la página se mostrara así. |

|  |
| --- |
| **PRACTICA NO. 3** |
| **NOMBRE: Estilos con CSS.** |
| **COMPETENCIA:** Dar diseño a una página web para que sea agradable a la vista. |
| **PROCEDIMIENTO:**    **Página web con la temperatura y el estado del sensor PIR**  client.println("<!DOCTYPE html>");  client.println("<html lang='es'>");  client.println("<head><meta charset='UTF-8'><title>Temperatura y Movimiento</title>");  **Empezaremos a darle estilo a la pagina we.**  **Abriremos la etiqueta estilo.**  client.println("<style>");  **Aquí le daremos estilo a las letras pequeñas.**  client.println("body { font-family: Arial, sans-serif; margin: 0; padding: 20px; background-color: #f4f4f4; color: #333; text-align: center; }");  **En esta parte le daremos estilo al título de la página.**  client.println("h1 { color: #3498db; }");  **Le daremos estilo a un contenedor donde se encontrarán los datos.**  client.println(".container { background-color: white; padding: 20px; border-radius: 8px; box-shadow: 0 0 10px rgba(0,0,0,0.1); margin: 20px auto; width: 60%; max-width: 600px; }");  **Daremos estilo a los botones.**  client.println("button { background-color: #3498db; color: white; border: none; padding: 10px 20px; font-size: 16px; border-radius: 5px; cursor: pointer; margin: 10px 5px; }");  client.println("button:hover { background-color: #2980b9; }");  client.println(".status { font-size: 18px; margin: 20px 0; }");  **Daremos estilo a las letras que anuncian que hay movimiento.**  client.println(".motion { color: #e74c3c; font-weight: bold; }");  **finalizamos cerrando la etiqueta estilo.**  client.println("</style>");  client.println("</head>");  client.println("<body>");  client.println("<div class='container'>");  client.println("<h1>Temperatura y Movimiento</h1>");  client.print("<p>La temperatura es de ");  client.print(t);  client.println(" °C.</p>");    **Mostrar el estado del sensor PIR**  if (pirState) {  client.println("<p class='status'>El sensor PIR está activado.</p>");  } else {  client.println("<p class='status'>El sensor PIR está desactivado.</p>");  }  **Botones para encender o apagar el sensor PIR**  client.println("<a href='/encender'><button>Encender Sensor PIR</button></a>");  client.println("<a href='/apagar'><button>Apagar Sensor PIR</button></a>");  **Mostrar si se detectó movimiento**  if (motionDetected && pirState) {  client.println("<p class='motion'>¡Movimiento detectado!</p>");  } else {  client.println("<p>No se ha detectado movimiento.</p>");  }  client.println("</div>");  client.println("</body>");  client.println("</html>");    break;  }  }  }  client.stop(); **Cerrar la conexión con el cliente**  }  } |

|  |
| --- |
| **PRACTICA NO. 4** |
| **NOMBRE: Creación de tabla de datos de la BD** |
| **COMPETENCIA:** Emplea el uso de tablas para un mejor orden. |
| **PROCEDIMIENTO:**  Ya con lo que se a obtenido en las anteriores practicas crearemos una tabla donde se concentraran los datos.    **Enviar la respuesta HTTP.**  client.println("HTTP/1.1 200 OK");  client.println("Content-Type: text/html");  client.println("Connection: close");  client.println();    **Página web con la temperatura y el estado del sensor PIR.**  client.println("<!DOCTYPE html>");  client.println("<html lang='es'>");  client.println("<head><meta charset='UTF-8'><title>Temperatura y Movimiento</title></head>");  client.println("<body>");  client.println("<h1>Temperatura y Movimiento</h1>");    **Mostrar los datos en una tabla sin estilo.**  Para esto le daremos un borde a la tabla y por cada dato abriremos una etiqueta que se cierra después.   * **<table>: Define la tabla.** * **<tr>: Define cada fila dentro de la tabla.** * **<th>: Define las celdas de encabezado (normalmente en negrita y centradas).** * **<td>: Define las celdas de datos en una fila.**   client.println("<table border='1'>");  client.println(" <tr> <th>Temperatura (°C) </th> <td> ");  client.print(t);  client.println("</td></tr>");    **Mostrar el estado del sensor PIR.**  client.println("<tr><th>Estado del Sensor PIR</th><td>");  if (pirState) {  client.println("Activado");  } else {  client.println("Desactivado");  }  client.println("</td></tr>");    **Mostrar si se detectó movimiento.**  client.println("<tr><th>Movimiento Detectado</th><td>");  if (motionDetected && pirState) {  client.println("Sí");  } else {  client.println("No");  }  client.println("</td></tr>");    client.println("</table>");    **Botones para encender o apagar el sensor PIR.**  client.println("<br><a href='/encender'><button>Encender Sensor PIR</button></a>");  client.println("<a href='/apagar'><button>Apagar Sensor PIR</button></a>");  client.println("</body>");  client.println("</html>");    break;  }  }  }  client.stop();  **Cerrar la conexión con el cliente**  }  }  **Resultado.** |
| **PRACTICA NO. 5** |
| **NOMBRE: Estilo a la tabla.** |
| **COMPETENCIA:** Aplica el estilo en tabla para dar una mejor apariencia. |
| **PROCEDIMIENTO:**  **Ya que tenemos la tabla hecha le daremos estilo.**    client.println("<!DOCTYPE html>");  client.println("<html lang='es'>");  client.println("<head><meta charset='UTF-8'><title>Temperatura y Movimiento</title>");  **De la misma forma que en la anterior practica abriremos la etiqueta de estilos.**  client.println("<style>");  **Todo esto será el estilo de la tabla.**  client.println("body { font-family: Arial, sans-serif; background-color: #f4f4f4; margin: 0; padding: 20px; }");  client.println("h1 { text-align: center; color: #3498db; }");  client.println("table { width: 60%; margin: 0 auto; border-collapse: collapse; }");  client.println("th, td { padding: 12px; text-align: left; border: 1px solid #ddd; }");  client.println("th { background-color: #3498db; color: white; font-weight: bold; }");  client.println("td { background-color: #ffffff; }");  **Fila de fondo alterno.**  client.println("tr:nth-child(even) td { background-color: #f2f2f2; }");  client.println("tr:hover { background-color: #d1e0e0; }");  **Al pasar el ratón este resaltar.**  client.println("button { background-color: #3498db; color: white; border: none; padding: 10px 20px; font-size: 16px; border-radius: 5px; cursor: pointer; margin: 10px 5px; }");  client.println("button:hover { background-color: #2980b9; }");  **De la misma forma que en la práctica anterior cerramos.**  client.println("</style>");  client.println("</head>");  client.println("<body>");  client.println("<h1>Temperatura y Movimiento</h1>");  **Llenado de la tabla.**  client.println("<table>");  client.println("<tr><th>Temperatura (°C)</th><td>");  client.print(t);  client.println("</td></tr>");  client.println("<tr><th>Estado del Sensor PIR</th><td>");  if (pirState) {  client.println("Activado");  } else {  client.println("Desactivado");  }  client.println("</td></tr>");  client.println("<tr><th>Movimiento Detectado</th><td>");  if (motionDetected && pirState) {  client.println("Sí");  } else {  client.println("No");  }  client.println("</td></tr>");  client.println("</table>");  client.println("<br><a href='/encender'><button>Encender Sensor PIR</button></a>");  client.println("<a href='/apagar'><button>Apagar Sensor PIR</button></a>");  client.println("</body>");  client.println("</html>");  **Resultado.** |

|  |
| --- |
| **PRACTICA NO. 6** |
| **NOMBRE: Conexión a xampp modelo, vista y controlador.** |
| **COMPETENCIA: crea diferentes carpetas de tu pagina web para su mejor administración.** |
| **PROCEDIMIENTO:**  1. Iniciar XAMPP  Primero, asegúrate de que los servicios de Apache y MySQL estén en ejecución desde el panel de control de XAMPP.  2. Crear una base de datos en MySQL  Inicia sesión (si es necesario) y crea una base de datos nueva desde la interfaz de phpMyAdmin.  3. Crear un archivo PHP para conectar con MySQL  Ahora, crea un archivo PHP para establecer la conexión con la base de datos. Guarda el siguiente código como “conectar.php” en la carpeta htdocs de XAMPP **(ubicada generalmente en C:\xampp\htdocs).**  #include <WiFi.h>  #include <HTTPClient.h>  #include <DHT.h>  **Recuerda que esta parte se configura con tu wifi y tu dirección.**  const char\* ssid = "prac6";  const char\* password = "000000000000";  const char\* serverName = "http://192.168.0.00/Practica6/modelo/guardar\_datos.php";  **Definir el pin al que está conectado el sensor DHT.**  #define DHTPIN 14  #define DHTTYPE DHT22 **Usa DHT11 si estás usando ese sensor.**  **Definir el pin para el sensor PIR.**  #define PIRPIN 13  **Crear objeto DHT para el sensor.**  DHT dht(DHTPIN, DHTTYPE);  **Variables para almacenar el estado del sensor PIR y la detección de movimiento.**  bool pirStatus = false;  bool movement = false;  void setup() {  **Inicializa la comunicación serial.**  Serial.begin(115200);  **Inicializa el sensor DHT.**  dht.begin();  **Configura el pin PIR como entrada.**  pinMode(PIRPIN, INPUT);  **Conectar a la red Wi-Fi.**  WiFi.begin(ssid, password);  **Espera hasta que se conecte a la red Wi-Fi.**  while (WiFi.status() != WL\_CONNECTED) {  delay(1000);  Serial.println("Intentando conectar a WiFi...");  }  Serial.println("Conectado a WiFi");  }  void loop() {  **Lee la temperatura desde el sensor DHT.**  float temperature = dht.readTemperature();  **Lee el estado del sensor PIR.**  pirStatus = digitalRead(PIRPIN);  **Verifica si el sensor PIR detecta movimiento.**  if (pirStatus) {  movement = true;  } else {  movement = false;  }  **Si la lectura de la temperatura falla, imprime un error y termina.**  if (isnan(temperature)) {  Serial.println("Error al leer el sensor DHT");  return;  }  **Verifica si el ESP32 está conectado a la red Wi-Fi.**  if (WiFi.status() == WL\_CONNECTED) {  HTTPClient http; **Crea el objeto HTTPClient.**  **Inicia la solicitud HTTP a la URL del servidor.**  http.begin(serverName);  http.addHeader("Content-Type", "application/x-www-form-urlencoded"); **Define el tipo de contenido.**  **Prepara los datos que se enviarán en la solicitud POST.**  String postData = "temperature=" + String(temperature) + "&pirStatus=" + String(pirStatus) + "&movement=" + String(movement);  **Envía los datos al servidor usando POST.**  int httpResponseCode = http.POST(postData);  **Imprime la respuesta del servidor o el código de error si la conexión falla.**  if (httpResponseCode > 0) {  Serial.println(httpResponseCode); **Imprime el código de respuesta HTTP.**  Serial.println(http.getString());  **Imprime la respuesta del servidor.**  } else {  Serial.print("Error en la conexión: ");  Serial.println(httpResponseCode); **Imprime el código de error si la conexión falla.**  }  **Finaliza la conexión HTTP.**  http.end();  } else {  **Si Wi-Fi no está conectado, intenta reconectar.**  Serial.println("WiFi desconectado, intentando reconectar...");  while (WiFi.status() != WL\_CONNECTED) {  delay(1000);  Serial.println("Intentando reconectar a WiFi...");  WiFi.begin(ssid, password);  }  Serial.println("Reconexión exitosa");  }  **Espera 15 segundos antes de realizar la siguiente lectura.**  delay(15000);  }  **Modelo vista controlador, son las carpetas donde tendremos cada parte de la pagina para facilitar su modificación.**    **Index.html**  <!DOCTYPE html>  <html lang="es">  <head>  <!-- Establecemos la codificación de caracteres a UTF-8 para soportar caracteres especiales -->  <meta charset="UTF-8">  <title>Temperatura y Movimiento</title>  <!-- Vinculamos un archivo de estilo CSS para dar formato a la página -->  <link rel="stylesheet"  **En esta parte mandaremos a llamar los estilos que se encuentran en la carpeta modelo.**  href="http://localhost/Practica6/modelo/style.css">  </head>  <body>  <!-- Título principal de la página -->  <h1>Temperatura y Movimiento</h1>  <!-- Tabla para mostrar los datos de temperatura, estado del sensor PIR y si hay movimiento -->  <table>  <tr>  <!-- Fila para mostrar la temperatura -->  <th>Temperatura (°C)</th>  <!-- Celda para mostrar la temperatura, inicialmente con el texto "Cargando..." -->  <td id="temperature">Cargando...</td>  </tr>  <tr>  <!-- Fila para mostrar la última actualización -->  <th>Última Actualización</th>  <!-- Celda para mostrar la fecha y hora de la última actualización -->  <td id="timestamp">Cargando...</td>  </tr>  <tr>  <!-- Fila para mostrar el estado del sensor PIR -->  <th>Estado del Sensor PIR</th>  <!-- Celda para mostrar si el sensor PIR está activado o desactivado -->  <td id="pirStatus">Cargando...</td>  </tr>  <tr>  <!-- Fila para mostrar si se detectó movimiento -->  <th>Movimiento Detectado</th>  <!-- Celda para mostrar si se detectó movimiento o no -->  <td id="movement">Cargando...</td>  </tr>  </table>  <br>  <!-- Botones para encender y apagar el sensor PIR -->  <button id="enablePIRBtn">Encender Sensor PIR</button>  <button id="disablePIRBtn">Apagar Sensor PIR</button>  <script>  **Función para obtener los datos desde el servidor.**  async function fetchData() {  try {  **Realizamos una solicitud GET a la URL que proporciona los datos.**  const response = await fetch('http://192.168.1.18/Practica6/modelo/guardar\_datos.php');    **Si la respuesta no es exitosa, lanzamos un error.**  if (!response.ok) {  throw new Error('Error en la solicitud');  }  **Convertimos la respuesta en formato JSON.**  const data = await response.json();  **Actualizamos el contenido de las celdas de la tabla con los datos obtenidos.**  document.getElementById('temperature').innerText = `${data.temperature} °C`;  document.getElementById('timestamp').innerText = data.timestamp;  document.getElementById('pirStatus').innerText = data.pirStatus ? 'Activado' : 'Desactivado';  document.getElementById('movement').innerText = data.movement ? 'Sí' : 'No';  } catch (error) {  **Si ocurre un error, mostramos un mensaje de error en lugar de los datos.**  document.getElementById('temperature').innerText = 'Error al obtener datos';  document.getElementById('timestamp').innerText = 'Error al obtener datos';  document.getElementById('pirStatus').innerText = 'Error al obtener datos';  document.getElementById('movement').innerText = 'Error al obtener datos';  console.error('Error:', error);  }  }  **Función para enviar comandos al servidor (encender o apagar el sensor PIR).**  async function sendCommand(command) {  try {  **Realizamos una solicitud GET para encender o apagar el sensor PIR.**  const response = await fetch(`http://192.168.1.18/${command}`);    **Si la respuesta no es exitosa, lanzamos un error.**  if (!response.ok) {  throw new Error('Error en la solicitud');  }  **Mostramos la respuesta del servidor en la consola.**  console.log(await response.text());  **Después de enviar el comando, actualizamos los datos en la página.**  fetchData();  } catch (error) {  **Si ocurre un error, lo mostramos en la consola.**  console.error('Error:', error);  }  }  **Añadimos eventos a los botones para enviar los comandos adecuados al hacer clic.**  document.getElementById('enablePIRBtn').addEventListener('click', () => sendCommand('encender'));  document.getElementById('disablePIRBtn').addEventListener('click', () => sendCommand('apagar'));  **Actualizamos los datos cada 15 segundos.**  setInterval(fetchData, 15000);  **Cuando se carga la página, obtenemos los primeros datos.**  fetchData();  </script>  </body>  </html>  **Style.css**  body { font-family: Arial, sans-serif; background-color: #f4f4f4; margin: 0; padding: 20px; }  h1 { text-align: center; color: #3498db; }  table { width: 60%; margin: 0 auto; border-collapse: collapse; }  th, td { padding: 12px; text-align: left; border: 1px solid #ddd; }  th { background-color: #3498db; color: white; font-weight: bold; }  td { background-color: #ffffff; }  tr:nth-child(even) td { background-color: #f2f2f2; } /\* Fila de fondo alterno \*/  tr:hover { background-color: #d1e0e0; } /\* Resaltar al pasar el ratón \*/  button { background-color: #3498db; color: white; border: none; padding: 10px 20px; font-size: 16px; border-radius: 5px; cursor: pointer; margin: 10px 5px; }  button:hover { background-color: #2980b9; }  **Resultado.** |
| **PRACTICA NO. 7** |
| **NOMBRE: registro de datos e impresión de datos en una tabla.** |
| **COMPETENCIA: Extraer datos de la base de datos en una tabla para su comprensión.** |
| **PROCEDIMIENTO:**  1. Iniciar XAMPP  Primero, asegúrate de que los servicios de Apache y MySQL estén en ejecución desde el panel de control de XAMPP.  2. Crear una base de datos en MySQL  Inicia sesión (si es necesario) y crea una base de datos nueva desde la interfaz de phpMyAdmin.  3. Crear un archivo PHP para conectar con MySQL  Ahora, crea un archivo PHP para establecer la conexión con la base de datos. Guarda el siguiente código como “conectar.php” en la carpeta htdocs de XAMPP **(ubicada generalmente en C:\xampp\htdocs).**  #include <WiFi.h>  #include <HTTPClient.h>  #include <DHT.h>  **Recuerda que esta parte se configura con tu wifi y tu dirección.**  const char\* ssid = "prac6";  const char\* password = "000000000000";  const char\* serverName = "http://192.168.0.00/Practica6/modelo/guardar\_datos.php";  **Definir el pin al que está conectado el sensor DHT.**  #define DHTPIN 14  #define DHTTYPE DHT22 **Usa DHT11 si estás usando ese sensor.**  **Definir el pin para el sensor PIR.**  #define PIRPIN 13  **Crear objeto DHT para el sensor.**  DHT dht(DHTPIN, DHTTYPE);  **Variables para almacenar el estado del sensor PIR y la detección de movimiento.**  bool pirStatus = false;  bool movement = false;  void setup() {  **Inicializa la comunicación serial.**  Serial.begin(115200);  **Inicializa el sensor DHT.**  dht.begin();  **Configura el pin PIR como entrada.**  pinMode(PIRPIN, INPUT);  **Conectar a la red Wi-Fi.**  WiFi.begin(ssid, password);  **Espera hasta que se conecte a la red Wi-Fi.**  while (WiFi.status() != WL\_CONNECTED) {  delay(1000);  Serial.println("Intentando conectar a WiFi...");  }  Serial.println("Conectado a WiFi");  }  void loop() {  **Lee la temperatura desde el sensor DHT.**  float temperature = dht.readTemperature();  **Lee el estado del sensor PIR.**  pirStatus = digitalRead(PIRPIN);  **Verifica si el sensor PIR detecta movimiento.**  if (pirStatus) {  movement = true;  } else {  movement = false;  }  **Si la lectura de la temperatura falla, imprime un error y termina.**  if (isnan(temperature)) {  Serial.println("Error al leer el sensor DHT");  return;  }  **Verifica si el ESP32 está conectado a la red Wi-Fi.**  if (WiFi.status() == WL\_CONNECTED) {  HTTPClient http; **Crea el objeto HTTPClient.**  **Inicia la solicitud HTTP a la URL del servidor.**  http.begin(serverName);  http.addHeader("Content-Type", "application/x-www-form-urlencoded"); **Define el tipo de contenido.**  **Prepara los datos que se enviarán en la solicitud POST.**  String postData = "temperature=" + String(temperature) + "&pirStatus=" + String(pirStatus) + "&movement=" + String(movement);  **Envía los datos al servidor usando POST.**  int httpResponseCode = http.POST(postData);  **Imprime la respuesta del servidor o el código de error si la conexión falla.**  if (httpResponseCode > 0) {  Serial.println(httpResponseCode); **Imprime el código de respuesta HTTP.**  Serial.println(http.getString());  **Imprime la respuesta del servidor.**  } else {  Serial.print("Error en la conexión: ");  Serial.println(httpResponseCode); **Imprime el código de error si la conexión falla.**  }  **Finaliza la conexión HTTP.**  http.end();  } else {  **Si Wi-Fi no está conectado, intenta reconectar.**  Serial.println("WiFi desconectado, intentando reconectar...");  while (WiFi.status() != WL\_CONNECTED) {  delay(1000);  Serial.println("Intentando reconectar a WiFi...");  WiFi.begin(ssid, password);  }  Serial.println("Reconexión exitosa");  }  **Espera 15 segundos antes de realizar la siguiente lectura.**  delay(15000);  }  **Index.html**  <!DOCTYPE html>  <html lang="es">  <head>  <!-- Establecemos la codificación de caracteres a UTF-8 para soportar caracteres especiales -->  <meta charset="UTF-8">  <title>Temperatura y Movimiento</title>  <!-- Vinculamos un archivo de estilo CSS para dar formato a la página -->  <link rel="stylesheet" href="http://localhost/Practica6/modelo/style.css">  </head>  <body>  <!-- Título principal de la página -->  <h1>Temperatura y Movimiento</h1>  <!-- Tabla para mostrar los datos de la base de datos -->  <table>  <thead>  <tr>  <th>Temperatura (°C)</th>  <th>Última Actualización</th>  <th>Estado del Sensor PIR</th>  <th>Movimiento Detectado</th>  </tr>  </thead>  <tbody id="data-table">  <!-- Las filas de los datos se llenarán mediante JavaScript -->  </tbody>  </table>  <br>  <!-- Botones para encender y apagar el sensor PIR -->  <button id="enablePIRBtn">Encender Sensor PIR</button>  <button id="disablePIRBtn">Apagar Sensor PIR</button>  <script>  **Función para obtener los datos desde el servidor**  async function fetchData() {  try {  **// Realizamos una solicitud GET a la URL que proporciona los datos de la base de datos**  const response = await fetch('http://192.168.1.18/Practica6/modelo/guardar\_datos.php');    **Si la respuesta no es exitosa, lanzamos un error**  if (!response.ok) {  throw new Error('Error en la solicitud');  }  **Convertimos la respuesta en formato JSON**  const data = await response.json();  **Llamamos a la función que llena la tabla con los datos obtenidos**  populateTable(data);  } catch (error) {  **Si ocurre un error, mostramos un mensaje de error en la consola**  console.error('Error al obtener datos:', error);  }  }  **Función para llenar la tabla con los datos de la base de datos**  function populateTable(data) {  const tableBody = document.getElementById('data-table');  tableBody.innerHTML = ''; // Limpiamos la tabla antes de llenarla  data.forEach(item => {  **Creamos una nueva fila para cada entrada de datos**  const row = document.createElement('tr');    **Creamos las celdas de cada fila con los datos correspondientes**  const tempCell = document.createElement('td');  tempCell.textContent = `${item.temperature} °C`;  row.appendChild(tempCell);  const timestampCell = document.createElement('td');  timestampCell.textContent = item.timestamp;  row.appendChild(timestampCell);  const pirStatusCell = document.createElement('td');  pirStatusCell.textContent = item.pirStatus ? 'Activado' : 'Desactivado';  row.appendChild(pirStatusCell);  const movementCell = document.createElement('td');  movementCell.textContent = item.movement ? 'Sí' : 'No';  row.appendChild(movementCell);  **Agregamos la fila a la tabla**  tableBody.appendChild(row);  });  }  **Función para enviar comandos al servidor (encender o apagar el sensor PIR)**  async function sendCommand(command) {  try {  **Realizamos una solicitud GET para encender o apagar el sensor PIR**  const response = await fetch(`http://192.168.1.18/${command}`);    **Si la respuesta no es exitosa, lanzamos un error**  if (!response.ok) {  throw new Error('Error en la solicitud');  }  **Mostramos la respuesta del servidor en la consola**  console.log(await response.text());  **Después de enviar el comando, actualizamos los datos en la página**  fetchData();  } catch (error) {  **Si ocurre un error, lo mostramos en la consola**  console.error('Error:', error);  }  }  **Añadimos eventos a los botones para enviar los comandos adecuados al hacer clic**  document.getElementById('enablePIRBtn').addEventListener('click', () => sendCommand('encender'));  document.getElementById('disablePIRBtn').addEventListener('click', () => sendCommand('apagar'));  **Actualizamos los datos cada 15 segundos**  setInterval(fetchData, 15000);  **Cuando se carga la página, obtenemos los primeros datos**  fetchData();  </script>  </body>  </html>  **Style.css**  body { font-family: Arial, sans-serif; background-color: #f4f4f4; margin: 0; padding: 20px; }  h1 { text-align: center; color: #3498db; }  table { width: 60%; margin: 0 auto; border-collapse: collapse; }  th, td { padding: 12px; text-align: left; border: 1px solid #ddd; }  th { background-color: #3498db; color: white; font-weight: bold; }  td { background-color: #ffffff; }  tr:nth-child(even) td { background-color: #f2f2f2; } /\* Fila de fondo alterno \*/  tr:hover { background-color: #d1e0e0; } /\* Resaltar al pasar el ratón \*/  button { background-color: #3498db; color: white; border: none; padding: 10px 20px; font-size: 16px; border-radius: 5px; cursor: pointer; margin: 10px 5px; }  button:hover { background-color: #2980b9; }  **Resultado.** |
| **PRACTICA NO. 8** |
| **NOMBRE: Conexión de sensor infrarrojo, movimiento y un potenciómetro.** |
| **CONPETENCIA:** Utiliza sensores para crear datos analógicos. |
| **PROCEDIMIENTO:**  1. Iniciar XAMPP  Primero, asegúrate de que los servicios de Apache y MySQL estén en ejecución desde el panel de control de XAMPP.  **Arduino.IDE**  #include <WiFi.h>  #include <HTTPClient.h>  const char\* ssid = "prueba"; // red Internet de tipo 2.4  const char\* password = "12345678";  const char\* serverName = "http://192.168.1.83/Wemos/modelo/guardar\_datos.php";  const int potPin = 36; **// Pin del potenciómetro**  const int irSensorPin = 14; **// Pin del sensor infrarrojo (SensorInfra)**  const int pirSensorPin = 17; **// Pin del sensor PIR (SensorProx)**  void setup() {  Serial.begin(115200);  **//declaramos las variables autilizar en el proyecto**  WiFi.begin(ssid, password);  **//definimos el tipo de señal que darán cada variable. En este caso los 3 sensores son de entrada de datos**  pinMode(potPin, INPUT);  pinMode(irSensorPin, INPUT);  pinMode(pirSensorPin, INPUT);  **//Metodo de conexion a la red conectada y clarada en la parte de arriba**  while (WiFi.status() != WL\_CONNECTED) {  delay(1000);  Serial.println("Intentando conectar a WiFi...");  }  Serial.println("Conectado a WiFi");  }  void loop() {  int potValue = analogRead(potPin); **// Leer el valor del potenciómetro en este caso uno de 10k**  int irValue = digitalRead(irSensorPin); **// Leer el estado del sensor infrarrojo**  int pirValue = digitalRead(pirSensorPin); **// Leer el estado del sensor PIR**  Serial.print("Valor del Potenciómetro: ");  Serial.println(potValue);  Serial.print("Sensor Infrarrojo (SensorInfra): ");  Serial.println(irValue == LOW ? "Detectado" : "No detectado");  Serial.print("Sensor PIR (SensorProx): ");  Serial.println(pirValue == LOW ? "Movimiento detectado" : "No se detecta movimiento");  if (WiFi.status() == WL\_CONNECTED) {  HTTPClient http;  http.begin(serverName);  http.addHeader("Content-Type", "application/x-www-form-urlencoded");  String postData = "Potenciometro=" + String(potValue) + "&SensorInfra=" + String(irValue) + "&SensorProx=" + String(pirValue);  int httpResponseCode = http.POST(postData);  if (httpResponseCode > 0) {  Serial.println(httpResponseCode);  Serial.println(http.getString());  } else {  Serial.print("Error en la conexión: ");  Serial.println(httpResponseCode);  }  http.end();  } else {  Serial.println("WiFi desconectado, intentando reconectar...");  while (WiFi.status() != WL\_CONNECTED) {  delay(1000);  Serial.println("Intentando reconectar a WiFi...");  WiFi.begin(ssid, password); // Intentar reconectar a WiFi  }  Serial.println("Reconexion exitosa");  }  delay(15000); **// Espera 15 segundos antes de leer y enviar el siguiente valor**  }  **Guardar\_datos.php**  <?php  **// Verificar si los valores están definidos en el POST**  if (isset($\_POST['Potenciometro']) && isset($\_POST['SensorInfra']) && isset($\_POST['SensorProx'])) {  $potValue = $\_POST['Potenciometro'];  $irValue = $\_POST['SensorInfra'];  $pirValue = $\_POST['SensorProx'];  $fecha = date('Y-m-d H:i:s'); // Obtener la fecha y hora actuales del servidor  **// Conectar a la base de datos**  $conexion = new mysqli("localhost", "root", "", "grafica");  if ($conexion->connect\_error) {  die("Conexión fallida: " . $conexion->connect\_error);  }  **// Preparar la consulta para evitar inyecciones SQL**  $stmt = $conexion->prepare("INSERT INTO sensores\_2 (Potenciometro, SensorInfra, SensorProx, fecha) VALUES (?, ?, ?, ?)");  $stmt->bind\_param("iiis", $potValue, $irValue, $pirValue, $fecha);  **// Ejecutar la consulta y verificar si fue exitosa**  if ($stmt->execute()) {  echo "Datos recibidos y guardados correctamente";  } else {  echo "Error al guardar los datos: " . $stmt->error;  }  **// Cerrar la conexión**  $stmt->close();  $conexion->close();  } else {  echo "No se recibió algún dato necesario";  }  **Tabla.php**  **Este archivo se colocara en la carpeta controlador.**  document.addEventListener("DOMContentLoaded", function() {  const url = "http://192.168.137.161/Vemos/modelo/crud.php";  async function fetchData() {  try {  const response = await fetch(url);  if (!response.ok) {  throw new Error('Error en la solicitud');  }  const data = await response.json();  populateTable(data);  } catch (error) {  console.error('Error:', error);  }  }  function populateTable(data) {  const tableBody = document.getElementById('data-table-body');  tableBody.innerHTML = '';  data.forEach(item => {  const row = document.createElement('tr');  const potentiometerCell = document.createElement('td');  potentiometerCell.textContent = item.Potenciometro;  row.appendChild(potentiometerCell);  const sensorInfraCell = document.createElement('td');  sensorInfraCell.textContent = item.SensorInfra;  row.appendChild(sensorInfraCell);  const sensorProxCell = document.createElement('td');  sensorProxCell.textContent = item.SensorProx;  row.appendChild(sensorProxCell);  const dateCell = document.createElement('td');  dateCell.textContent = item.fecha;  row.appendChild(dateCell);  const actionCell = document.createElement('td');  const editButton = document.createElement('button');  editButton.textContent = 'Actualizar';  editButton.addEventListener('click', () => updateData(item));  actionCell.appendChild(editButton);  const deleteButton = document.createElement('button');  deleteButton.textContent = 'Borrar';  deleteButton.addEventListener('click', () => deleteData(item.id));  actionCell.appendChild(deleteButton);  row.appendChild(actionCell);  tableBody.appendChild(row);  });  }  async function saveData(potenciometro, sensorInfra, sensorProx) {  const formData = new FormData();  formData.append('Potenciometro', potenciometro);  formData.append('SensorInfra', sensorInfra);  formData.append('SensorProx', sensorProx);  try {  const response = await fetch(url, {  method: 'POST',  body: formData  });  if (!response.ok) {  throw new Error('Error en la solicitud');  }  alert('Datos guardados correctamente');  fetchData();  } catch (error) {  console.error('Error:', error);  }  }  async function updateData(item) {  const newPotenciometro = prompt('Nuevo valor para Potenciometro:', item.Potenciometro);  const newSensorInfra = prompt('Nuevo valor para Sensor Infra:', item.SensorInfra);  const newSensorProx = prompt('Nuevo valor para Sensor Prox:', item.SensorProx);  if (newPotenciometro !== null && newSensorInfra !== null && newSensorProx !== null) {  const putData = id=${item.id}&Potenciometro=${newPotenciometro}&SensorInfra=${newSensorInfra}&SensorProx=${newSensorProx};  try {  const response = await fetch(url, {  method: 'PUT',  headers: { 'Content-Type': 'application/x-www-form-urlencoded' },  body: putData  });  if (!response.ok) {  throw new Error('Error en la solicitud');  }  alert('Datos actualizados correctamente');  fetchData();  } catch (error) {  console.error('Error:', error);  }  }  }  async function deleteData(id) {  const deleteData = id=${id};  try {  const response = await fetch(url, {  method: 'DELETE',  headers: { 'Content-Type': 'application/x-www-form-urlencoded' },  body: deleteData  });  if (!response.ok) {  throw new Error('Error en la solicitud');  }  alert('Datos eliminados correctamente');  fetchData();  } catch (error) {  console.error('Error:', error);  }  }  document.getElementById('saveButton').addEventListener('click', () => {  const potenciometro = prompt('Valor para Potenciometro:');  const sensorInfra = prompt('Valor para Sensor Infra:');  const sensorProx = prompt('Valor para Sensor Prox:');  if (potenciometro !== null && sensorInfra !== null && sensorProx !== null) {  saveData(potenciometro, sensorInfra, sensorProx);  }  });  fetchData();  });  **Para poder llamar la tabla crearemos in index.php donde se llama el archivo de la tabla.**  <!DOCTYPE html>  <html lang="es">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Gráfica de Líneas y Datos de Sensores</title>  <link rel="stylesheet" href="../modelo/style.css">  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>  <script src="https://cdn.jsdelivr.net/npm/date-fns@2.27.0/dist/date-fns.min.js"></script>  <script src="https://cdn.jsdelivr.net/npm/chartjs-adapter-date-fns"></script>  </head>  <body>  <h1>Gráfica de Líneas y Datos de Sensores</h1>  <canvas id="graficaLineas"></canvas>  <h2>Datos de Sensores</h2>  <button id="saveButton">Guardar Datos</button>  <table border="1">  <thead>  **Le damos estructuraa la table.**  <tr>  <th>Potenciometro</th>  <th>Sensor Infra</th>  <th>Sensor Prox</th>  <th>Fecha</th>  <th>Acciones</th>  </tr>  </thead>  **Y aqui llamamos a la table con los datos.**  <tbody id="data-table-body">  <!-- Las filas de datos serán agregadas aquí por JavaScript -->  </tbody>  </table>  <script src="../controlador[\_{{{CITATION{{{\_1{](https://github.com/la9una/web/tree/ba1073ae044ebb7b538a3b13f0f9598f7c410bb6/docs%2Fbootstrap%2Falignci.md)  **Resultado.**  **Vista de xampp.**    **Vista de la pagina.** |
| **PRACTICA NO. 9** |
| **NOMBRE: Graficación de datos.** |
| **COMPETENCIA:** Crear una gráfica para la visualización de datos. |
| **PROCEDIMIENTO:**  1. Iniciar XAMPP  Primero, asegúrate de que los servicios de Apache y MySQL estén en ejecución desde el panel de control de XAMPP.  **En la misma carpeta de controlador crearemos un archivo llamado grafica.php donde colocaremos el código que graficara todos los valores obtenidos.**  async function obtenerDatos() {  try {  const respuesta = await fetch('http://192.168.137.161/Vemos/modelo/obtener\_datos.php', {  method: 'GET',  headers: {  'Content-Type': 'application/json'  }  });  if (!respuesta.ok) {  throw new Error(Error en la solicitud: ${respuesta.status});  }  const datos = await respuesta.json();  console.log('Datos recibidos:', datos); // Mensaje de depuración  return datos;  } catch (error) {  console.error('Error al obtener datos:', error); // Imprimir errores de solicitud  }  }  async function actualizarGrafica(grafica, etiquetas, valoresPotenciometro, valoresSensorInfra, valoresSensorProx) {  const datos = await obtenerDatos();  const nuevasEtiquetas = datos.map(d => new Date(d.fecha));  const nuevosValoresPotenciometro = datos.map(d => d.Potenciometro);  const nuevosValoresSensorInfra = datos.map(d => d.SensorInfra);  const nuevosValoresSensorProx = datos.map(d => d.SensorProx);  etiquetas.length = 0; // Limpiar el array de etiquetas  valoresPotenciometro.length = 0; // Limpiar el array de valores del potenciómetro  valoresSensorInfra.length = 0; // Limpiar el array de valores del sensor infrarrojo  valoresSensorProx.length = 0; // Limpiar el array de valores del sensor PIR  etiquetas.push(...nuevasEtiquetas); // Agregar nuevas etiquetas  valoresPotenciometro.push(...nuevosValoresPotenciometro); // Agregar nuevos valores del potenciómetro  valoresSensorInfra.push(...nuevosValoresSensorInfra); // Agregar nuevos valores del sensor infrarrojo  valoresSensorProx.push(...nuevosValoresSensorProx); // Agregar nuevos valores del sensor PIR  grafica.update(); // Actualizar la gráfica  }  async function crearGraficaLineas() {  const datos = await obtenerDatos();  const etiquetas = datos.map(d => new Date(d.fecha));  const valoresPotenciometro = datos.map(d => d.Potenciometro);  const valoresSensorInfra = datos.map(d => d.SensorInfra);  const valoresSensorProx = datos.map(d => d.SensorProx);  const ctx = document.getElementById('graficaLineas').getContext('2d');  const grafica = new Chart(ctx, {  type: 'line',  data: {  labels: etiquetas,  datasets: [  {  label: 'Valores Potenciómetro',  data: valoresPotenciometro,  backgroundColor: 'rgba(153, 102, 255, 0.2)',  borderColor: 'rgba(153, 102, 255, 1)',  borderWidth: 1  },  {  label: 'Valores Sensor Infra',  data: valoresSensorInfra,  backgroundColor: 'rgba(255, 99, 132, 0.2)',  borderColor: 'rgba(255, 99, 132, 1)',  borderWidth: 1  },  {  label: 'Valores Sensor Prox',  data: valoresSensorProx,  backgroundColor: 'rgba(75, 192, 192, 0.2)',  borderColor: 'rgba(75, 192, 192, 1)',  borderWidth: 1  }  ]  },  options: {  plugins: {  legend: {  display: true  }  },  scales: {  x: {  type: 'time',  time: {  unit: 'minute',  tooltipFormat: 'dd/MM/yyyy HH:mm',  displayFormats: {  minute: 'dd/MM/yyyy HH:mm'  }  },  title: {  display: true,  text: 'Fecha y Hora'  },  ticks: {  source: 'auto',  major: {  enabled: true  },  autoSkip: true,  maxTicksLimit: 10  }  },  y: {  beginAtZero: true  }  }  }  });  setInterval(() => actualizarGrafica(grafica, etiquetas, valoresPotenciometro, valoresSensorInfra, valoresSensorProx), 2000); // Actualizar cada 2 segundos  }  crearGraficaLineas();  **En el documento donde llamamos a la tabla llamaremos también a la grafica.**  <!DOCTYPE html>  <html lang="es">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Gráfica de Líneas y Datos de Sensores</title>  <link rel="stylesheet" href="../modelo/style.css">  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>  <script src="https://cdn.jsdelivr.net/npm/date-fns@2.27.0/dist/date-fns.min.js"></script>  <script src="https://cdn.jsdelivr.net/npm/chartjs-adapter-date-fns"></script>  </head>  <body>  <h1>Gráfica de Líneas y Datos de Sensores</h1>  **Esta sera la línea que llamara la grafica para ser visualizada.**  <canvas id="graficaLineas"></canvas>  <h2>Datos de Sensores</h2>  <button id="saveButton">Guardar Datos</button>  <table border="1">  <thead>  <tr>  <th>Potenciometro</th>  <th>Sensor Infra</th>  <th>Sensor Prox</th>  <th>Fecha</th>  <th>Acciones</th>  </tr>  </thead>  <tbody id="data-table-body">  <!-- Las filas de datos serán agregadas aquí por JavaScript -->  </tbody>  </table>  <script src="../controlador[\_{{{CITATION{{{\_1{](https://github.com/la9una/web/tree/ba1073ae044ebb7b538a3b13f0f9598f7c410bb6/docs%2Fbootstrap%2Falignci.md)  **Resultado.** |