ICETEC

REPORTE TÉCNICO

IOT - ICETEC

ESTUDIANTE:	MATRÍCULA
CLARISSA PEÑA	**A00840632**
LORAIN GONZÁLEZ	**A01707279**
GABRIEL LASCURAIN	**A01707236**
AKSEL DENEKEN	**A01711966**

Índice

CONFIGURACIÓN DE HADWARE

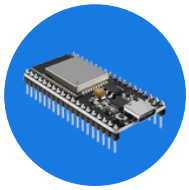
FUENTE DE ENERGÍA
CIRCUITO
MODELO 3D
ARDUINO
PLACA DE PELTIER
COSTOS

ICETEC

Componentes Técnicos

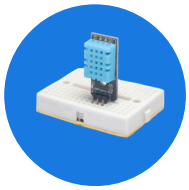
Para nuestro proyecto utilizamos los siguientes componentes técnicos para configurar el prototipo de hardware.

Componentes:



- **ESP32**

Es un microcontrolador potente con Wi-Fi y Bluetooth, ideal para proyectos IoT. Integra múltiples pines GPIO, procesador dual-core, y soporte para sensores y periféricos, permitiendo conectividad y automatización.

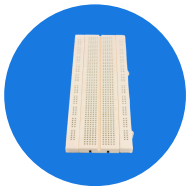


- **DTH11 (Sensor de humedad y temperatura)**

Es un sensor económico que mide temperatura y humedad. Utiliza una salida digital para enviar datos precisos a microcontroladores.

- **LEDs**

Es un componente electrónico que nos permite comprobar la señal y recibir respuestas de nuestro servidor para conocer el estatus de nuestro sistema, funcionan como indicadores de nuestro proyecto.



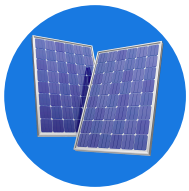
- **Protoboard**

Funge como el esqueleto de nuestro poryecto, nos permite conectar los componentes a poner a prueba y verificar su funcionamiento. Nos da una perspectiva de cómo debería de estar planteado nuestro circuito para pensar en un diseño más elegnte como una placa personalizada.

Componentes Técnicos

Para un funcionamiento final del proyecto, agregaríamos los siguientes componentes que nos ayudan a completar la idea del proyecto final.

Componentes:

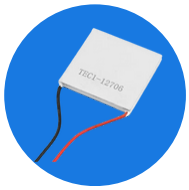


- **Panel solar monocristalino de 50W-100W**

Es un componente clave en el sistema de refrigeración autónoma, proporcionando energía solar eficiente y sostenible para su funcionamiento. Con una alta eficiencia de hasta el 20%, es capaz de generar electricidad en condiciones de luz no óptimas, ideal para zonas rurales o remotas sin acceso a la red eléctrica. Su diseño compacto y resistente asegura un bajo mantenimiento y una vida útil de más de 25 años. Este panel es perfecto para alimentar el sistema de refrigeración ICETEC, garantizando su operación autónoma y ecológica, con un bajo costo operativo y un rendimiento fiable incluso en condiciones climáticas adversas.

- **Puente de energía**

Este dispositivo regula la energía proveniente del panel para evitar fluctuaciones o sobrecarga.



- **Placa del Peltier de 40mm x 40mm con 12V**

Es un componente clave en el sistema de refrigeración autónoma. Utiliza el efecto termoeléctrico para transferir calor de un lado al otro, manteniendo una temperatura constante en el sistema de conservación de alimentos. Su bajo consumo energético la hace ideal para funcionar con energía solar, garantizando una solución sostenible. Además, su tamaño compacto y bajo mantenimiento la convierten en una opción eficiente y duradera para sistemas autónomos. Es escalable, permitiendo ajustarse a diferentes necesidades de refrigeración, y asegura un rendimiento fiable durante años.

Diagrama de configuración

ICETEC

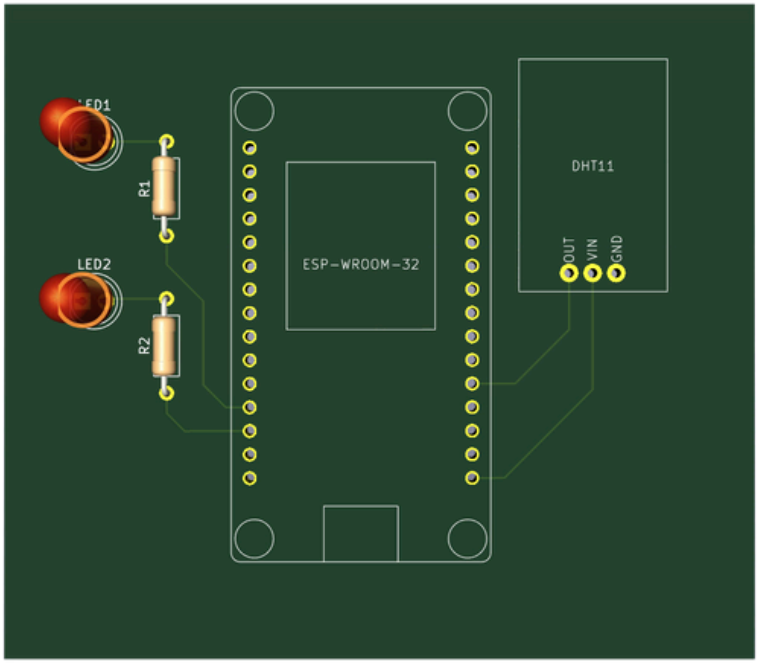
Costos de Hardware

Este es el presupuesto de nuestro proyecto a partir de los componentes empleados en el Hardware, ya que todos los programas utilizamos para la interconexión del proyecto son de licencia libre.

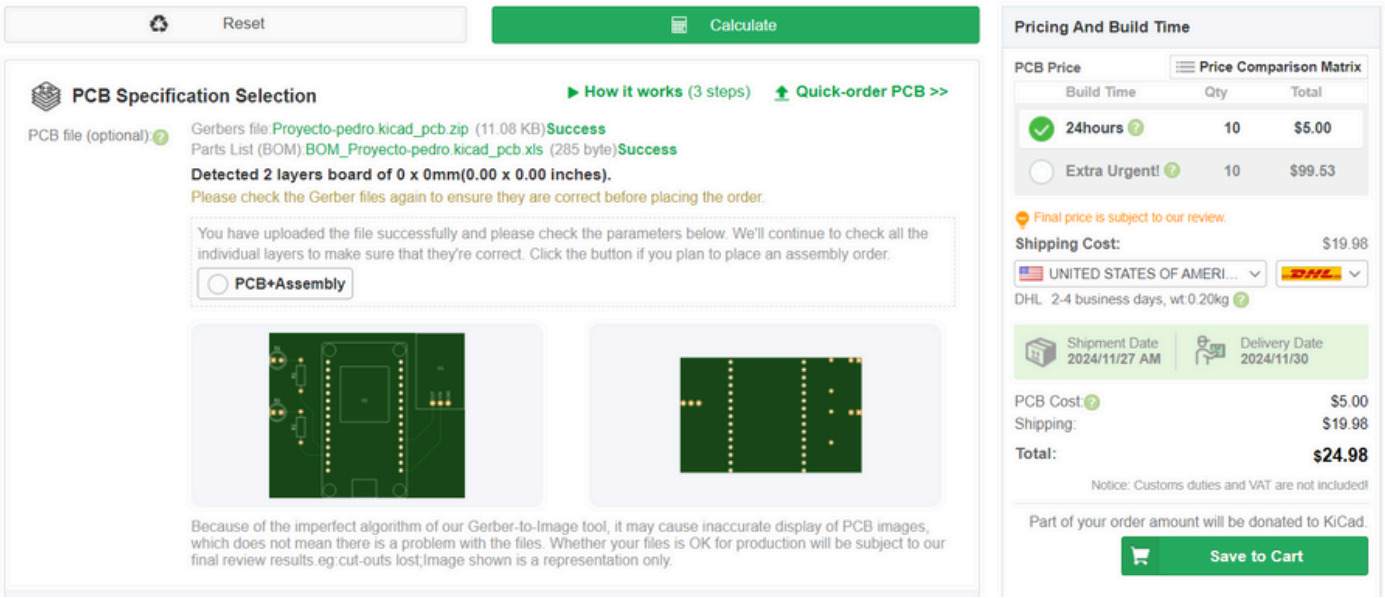
Componente	Descripción	Costo aproximado (MXN)
ESP32	ESP-WROOM-32-DEVKITV1	$150 - $250
DHT11	DHT11 module Board	$50 - $80
LEDs	2 LED 5mm	$2 - $5 por unidad
Protoboard	Protoboard 509-015 Steren	$40 - $80
Resistencias	2 de 220 Ohm	$1 - $2 por unidad
Cables Jumper	4 Macho/Macho y 5 Macho/Hembra	$40 - $60 por paquete de 40 cables

Modelo en 3D

Realizamos un proyecto de prototipo en Kicad con una placa donde podríamos simplificar la conexión y hacer más presentable nuestro proyecto, importando algunas librerías y realizando los modelos personalizados para nuestro sensor DHT11 y nuestra placa ESP-WROOM-32..



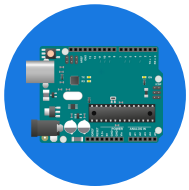
Obtenemos un presupuesto de mandar a hacer las placas sobre nuestro mismo modelo, profesionalmente a PCB.



Código de Arduino

Para el funcionamiento de la ESP32, configuramos el siguiente código que nos ayuda a darle a nuestro microprocesador las instrucciones que deberá ejecutar.

Codificación en Arduino:



- **Arduino**

Es una plataforma de hardware y software de código abierto diseñada. Consiste en placas con microcontroladores y un entorno de desarrollo integrado (IDE) que permite programarlas en C. En este caso, usamos ESP32, que es una placa de desarrollo avanzada de la familia Arduino con capacidades de WiFi y Bluetooth.

- **Librerías**
- **WiFi.h**: permite la conexión a redes WiFi.
- **HTTPClient.h**: gestiona solicitudes HTTP (GET/POST).
- **Arduino_JSON.h**: facilita el manejo de datos en formato JSON.
- **DHT.h**: controla el sensor de temperatura y humedad DHT11.

- **Puntos esenciales**

Seguimiento por pasos de nuestro código para llevar a cabo la comunicación.

- **Designación de pines:** definimos los pines de la ESP-32 para cada componente que conectaremos.

- **Conexión a internet:** le brindamos al código el ID y contraseña de nuestra red y probamos la conexión, si no se logra la placa se resetea cada 20 segundos.

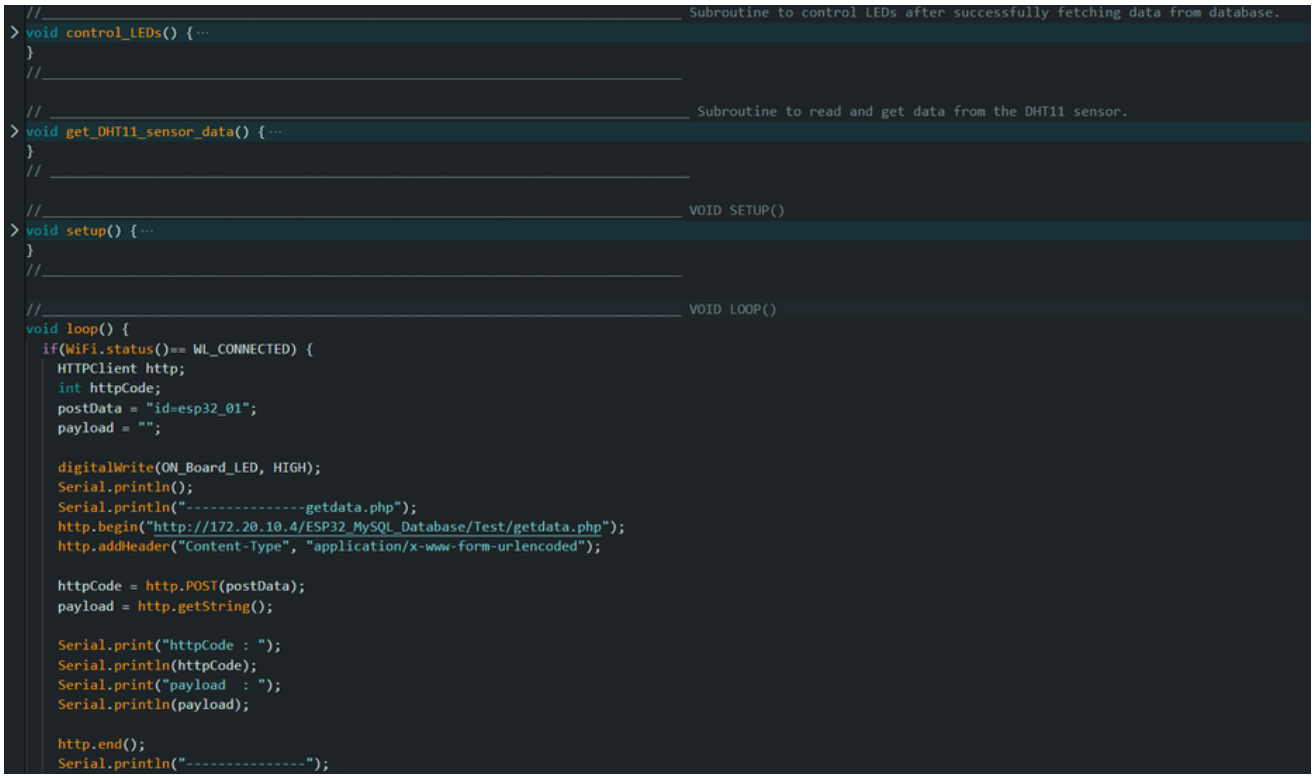
- **Control de LEDs:** el código toma e interpreta los JSON desde la base de datos para definir el estado de los LEDS, siendo encendido o apagado.

- **Lecturas del sensor DHT11:** utiliza las funciones de la libreria propia de nuestro sensor para realizar las lecturas, para posteriormente validarlos y establecer un estado de SUCCEED o FAILED dependiendo de si la lectura es válida o no.

Código de Arduino

- **Interacción con el servidor:**
  a. Solicitamos el estado de los LEDs en un POST de tipo JSON al getdata.php
  b. Actualizamos los datos del sensor y los LEDs, haciendo una solicitud al updateDHT11data_and_recordtable.php, enviando los datos nuevos a la base de datos.

- **Bucle principal:** es lo que nuestra placa estará realizando en bucle, comprueba la conexión de WiFi y procede a ciclar las funciones para controlar LEDs, leer el sensor y enviar datos al servidor.



```
//_____ Subroutine to control LEDs after successfully fetching data from database.
void control_LEDs() {···
}
//_____

// _____ Subroutine to read and get data from the DHT11 sensor.
void get_DHT11_sensor_data() {···
}
// _____

//_____ VOID SETUP()
void setup() {···
}
//_____

//_____ VOID LOOP()
void loop() {
  if(WiFi.status()== WL_CONNECTED) {
    HTTPClient http;
    int httpCode;
    postData = "id=esp32_01";
    payload = "";

    digitalWrite(ON_Board_LED, HIGH);
    Serial.println();
    Serial.println("---------------getdata.php");
    http.begin("http://172.20.10.4/ESP32_MySQL_Database/Test/getdata.php");
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");

    httpCode = http.POST(postData);
    payload = http.getString();

    Serial.print("httpCode : ");
    Serial.println(httpCode);
    Serial.print("payload  : ");
    Serial.println(payload);

    http.end();
    Serial.println("---------------");
```

BASE DE DATOS

MODELO DE TABLAS
RECOPILACIÓN DE DATOS
INTERCONEXIONES DEL PROYECTO

ICETEC

Base de datos

Nuestra DB está basada en el almacenamiento de datos obtenidos del sensor DTH11 y el estado de los LED controlados por una placa ESP32. Aquí te explico cómo estructuramos las tablas.

- **Tabla: esp32_table_dht11_leds_update:** Guarda los datos más recientes obtenidos del sensor DHT11 y el estado de los LED.

Campo	Tipo de Dato	Descripción
id	VARCHAR(255)	Identificador único de la placa
temperature	FLOAT(10,2)	Lectura de temperatura del sensor DHT11.
humidity	INT(3)	Lectura de humedad del sensor DHT11.
status_read_sensor_dht11	VARCHAR(255)	Estado de la lectura del sensor (ej. SUCCESS).
LED_01	VARCHAR(255)	Estado del LED 1 (ej. ON, OFF).
LED_02	VARCHAR(255)	Estado del LED 2 (ej. ON, OFF).
time	TIME	Hora de la lectura.
date	DATE	Fecha de la lectura.

Base de datos

- **Tabla: esp32_table_dht11_leds_record:** Guarda un historial completo de las lecturas de los sensores y los estados de los LEDs.

Relaciones: Puede relacionarse con la tabla de actualizaciones (esp32_table_dht11_leds_update) mediante el campo id.
Campos: Los mismos que esp32_table_dht11_leds_update, pero con un campo adicional board para identificar más detalles del hardware.
Ahora una vez creada nuestra base de datos , necesitamos comenzar a llenarla con la información correspondiente, para ellos se utilizaron las siguientes herramientas:

Herramientas:

- **HTTP (Hypertext Transfer Protocol)**

Es un protocolo de comunicación que permite la transferencia de información entre un cliente (como la ESP32) y un servidor (en este caso, Apache en XAMPP). Es el lenguaje común que ambos entienden para intercambiar datos.

- **XAMPP**

Es un paquete de software que convierte tu computadora en un servidor local para desarrollar y probar aplicaciones web. Incluye herramientas esenciales como Apache (servidor web) y MySQL (gestor de bases de datos), facilitando la creación de sistemas interactivos.

- **APACHE**

Es un servidor web que actúa como intermediario entre los clientes (como la ESP32) y el servidor local. Su función principal es recibir solicitudes HTTP, procesarlas, en nuestro proyecto recibe los datos enviados por la ESP32 mediante solicitudes HTTP y los dirige al archivo PHP

Base de datos

- **DataBase (Archivo de PHP)**

Nuestro archivo DataBase contiene la clase Database que actúa como un puente confiable entre el servidor web (Apache) y la base de datos (MySQL), permitiendo almacenar de manera eficiente los datos enviados por la ESP32.

¿Cómo lo hace?

Define los parámetros de conexión: Almacenan el nombre de la base de datos, el host (servidor), el usuario y la contraseña necesarios para conectar con MySQL.

Establece una conexión única: Crea una conexión a la base de datos utilizando PDO, que es un enfoque seguro y flexible para interactuar con bases de datos en PHP

Procesamiento de los datos: Cuando la ESP32 envía datos al servidor a través de HTTP, el archivo PHP que los procesa llama a Database::connect() para obtener una conexión activa a MySQL y guardar los datos en la base de datos.

- **MySQL y PHP MYADMIN**

MySQL es un sistema de gestión de bases de datos relacional que organiza y almacena información en tablas, permitiendo consultas rápidas y eficientes.

¿Cómo lo hace?

Cuando la ESP32 envía información al servidor, el archivo PHP que la recibe conecta con MySQL utilizando las credenciales configuradas. Luego, ejecuta comandos SQL para insertar los datos en las tablas correspondientes.

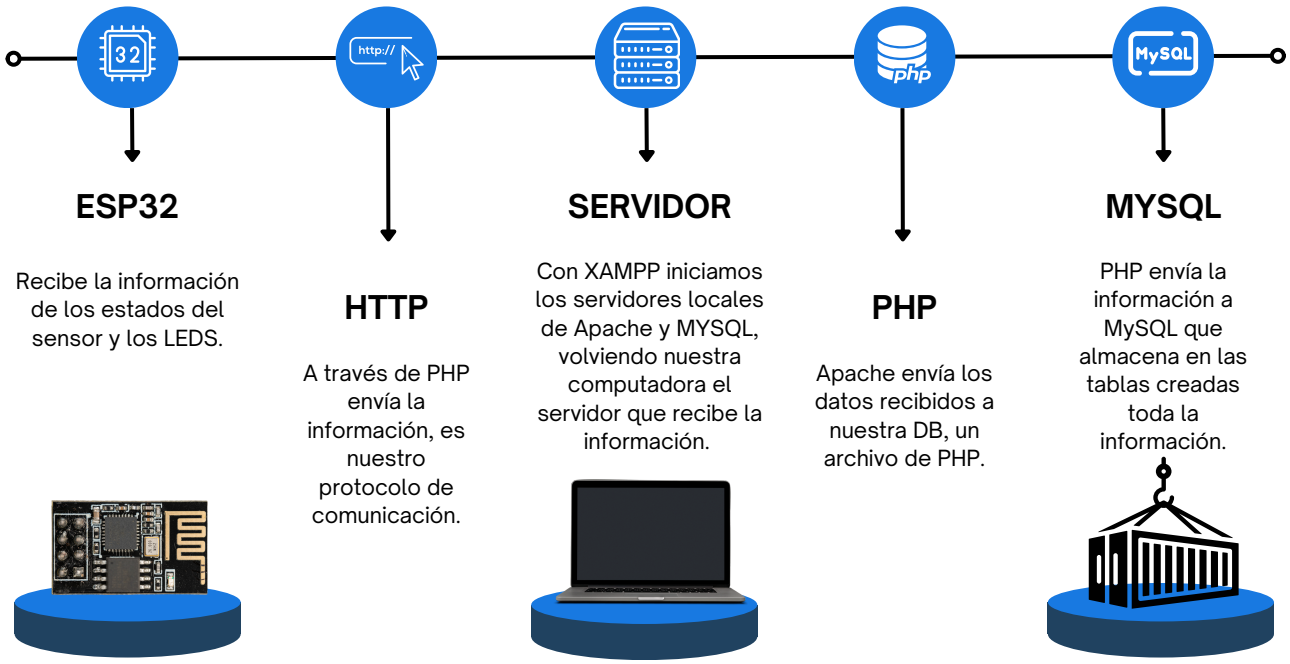
Nosotros utilizando la herramienta phpMyAdmin, accedemos a la base de datos para consultar, filtrar o analizar los datos guardados. Esto te permite verificar si las mediciones enviadas por la ESP32 se han registrado correctamente

Base de datos

¿Cómo funciona?
La ESP32 es un mensajero que lleva la información. En este caso, la información son los datos de temperatura, humedad, y el estado de los LEDs.
Para enviar los datos, la ESP32 utiliza Wi-Fi, como si mandara un mensaje por Internet. Este mensaje llega a un servidor, que en nuestro caso es la computadora, donde está la base de datos (LocalHost). Una vez ahí ordena los datos con la tabla
Para ello utilizamos las herramientas que se explican a continuación con un diagrama:

BASE DE DATOS - CONEXIONES

Demostración de cómo funciona la conexión de nuestra base de datos



ICETEC

Interconexión

Una vez tenemos almacenados los datos en nuestras tablas, necesitamos manipularlos para crear una interfaz o realizar modificaciones desde la web. Para ello se utilizaron las siguientes herramientas que nos ayudan hacer posible está interconexión del proyecto.

- **GETDATA (Archivo de PHP)**

Este código se encarga de recuperar datos específicos desde la base de datos, basándose en un identificador único (id) enviado mediante una solicitud HTTP (por POST o GET), y devolver los datos en formato JSON.

¿Cómo lo hace?

- **Incluye la conexión a la base de datos:** Usa include 'database.php'; para acceder a la clase Database, que establece la conexión con la base de datos.
- **Recepción de solicitudes**: Este archivo es llamado cuando se necesita recuperar datos específicos de la base de datos, como el estado de sensores o LEDs, mediante un id.

Su funcionamiento se maneja de la siguiente manera:

1. Se recibe una solicitud con un id por HTTP (POST o GET).
2. El servidor consulta la base de datos usando el id.
3. Si los datos existen, se devuelven en formato JSON. Si no, se envía un error.

- **updateDTH11data_and_recordtable (Archivo de PHP)**

Este código se usa para recibir datos actualizados de la ESP32, como la temperatura y humedad o los estados de los LEDs. La información se actualiza en la base de datos para reflejar los valores más recientes.

¿Cómo lo hace?

- **Recibe los datos enviados por la solicitud POST:** Se extraen varios valores del $_POST *(como id, temperature, humidity, status_read_sensor_dht11, led_01, led_02),* que son enviados
- **Actualiza los datos en la base de datos:** Realiza una actualización (UPDATE) en la tabla esp32_table_dht11_leds_update, que almacena los datos más recientes del sensor y el estado de los LEDs.
- **Genera un nuevo ID único para el registro:** Se genera un nuevo ID único mediante la función generate_string_id(10). Este ID será utilizado para registrar los datos en una tabla secundaria llamada esp32_table_dht11_leds_record.

Interconexión

- **Registra los datos en la tabla secundaria:** Una vez generado un ID único, se inserta un nuevo registro en la tabla esp32_table_dht11_leds_record. Esta tabla almacena todos los registros históricos de los datos del sensor DHT11 y el estado de los LEDs.

Este código asegura que los datos del sensor y el estado de los LEDs se actualicen en tiempo real, mientras que también mantiene un registro histórico para su análisis posterior.

- **updateLEDs (Archivo de PHP)**

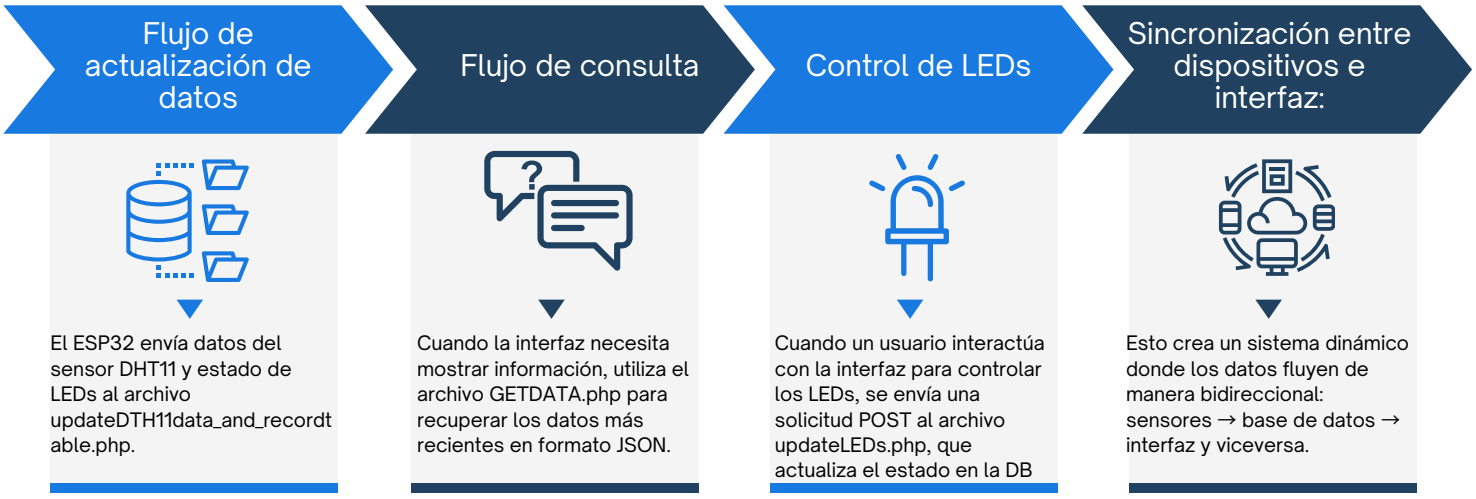
Es una parte esencial del control remoto de los LEDs en el proyecto. Permite recibir una solicitud (desde la interfaz de usuario) para cambiar el estado de un LED.

¿Cómo lo hace?

- **Control de LEDs:** Cuando se presiona el botón, la interfaz de usuario envía una solicitud POST con el id del dispositivo, el número de LED a actualizar (lednum), y el estado del LED (ledstate).
- **Actualización en tiempo real:** El código interactúa con la base de datos, actualizando el estado de los LEDs en la tabla esp32_table_dht11_leds_update.

DIAGRAMA - INTERCONEXIONES

Demostración de cómo funciona la interconexión del proyecto



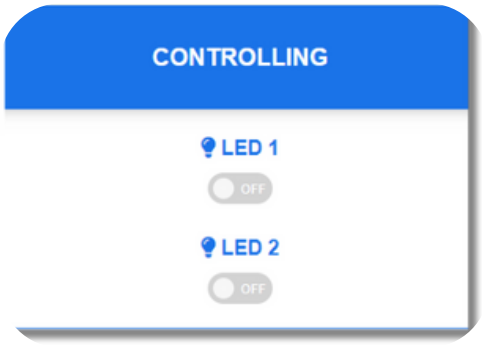
INTERFAZ

PÁGINA WEB

Nuestra interfaz web está diseñada para proporcionar una experiencia de usuario intuitiva y funcional para monitorear y controlar el sistema IoT de ICETEC. La interfaz permite visualizar los datos del sensor DHT11, el estado de los LEDs y acceder al historial de registros.

Diseño de página web

- Incluye un panel principal donde se muestran las lecturas del sensor de temperatura y humedad (DHT11) y el estado de los LEDs.
- Botones interactivos permiten controlar el encendido y apagado de los LEDs.
- Lectura en tiempo real: Los datos de temperatura y humedad se actualizan automáticamente desde la base de datos utilizando solicitudes GET.
- Control de LEDs: Los usuarios pueden encender o apagar los LEDs enviando comandos mediante solicitudes POST desde la página.



Cómo usar la interfaz

La página está alojada en un servidor local utilizando XAMPP, que permite conectar la base de datos al microcontrolador ESP32. Se puede acceder a la página web desde un navegador ingresando la dirección del servidor.

En el panel principal, se visualizan las lecturas de temperatura y humedad junto con el estado actual de los LEDs.
Los botones de la interfaz permiten al usuario controlar cada LED de forma independiente.

Nuestra interfaz se basa en 2 archivos: home.phb y recordtable.php

home.php

El archivo home.php es la puerta de entrada para los usuarios que interactúan con el sistema a través de una interfaz web. Su función principal es generar una página dinámica que permita monitorear los datos del sensor DHT11 (temperatura y humedad) y controlar los LEDs conectados al ESP32. Aquí hay una explicación detallada:

recordtable.php

El archivo recordtable.php es una pieza clave en la arquitectura del sistema, ya que actúa como intermediario entre la interfaz, la base de datos y el microcontrolador ESP32. Su función principal es gestionar las solicitudes para registrar y actualizar datos de sensores y LEDs en la base de datos, así como mantener un historial de las interacciones. Este archivo recibe las solicitudes POST desde la interfaz, actualiza el estado de los LEDs en la base de datos y genera un nuevo registro en la tabla de registro. También procesa las solicitudes GET para devolver datos en formato JSON.

El ESP32 envía datos del sensor y estado de los LEDs al archivo recordtable.php. Cuando el usuario interactúa con los botones, home.php envía comandos que son procesados por recordtable.php para actualizar la base de datos y el estado de los LEDs.

Líneas de código principales

```
include 'database.php';
$conn = Database::connect();
```

Se asegura una conexión segura y eficiente con la base de datos.

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $ledId = $_POST['lednum'];
    $ledState = $_POST['ledstate'];
    // Código para actualizar el estado del LED en la base de datos
}
```

Este código procesa los cambios realizados por el usuario en la interfaz.

```php
$query = "SELECT * FROM esp32_table_dht11_leds_update OR
$result = $conn->query($query);
echo json_encode($result->fetch_assoc());
```

Recupera los datos más recientes para mostrar en la página web.

Interfaz