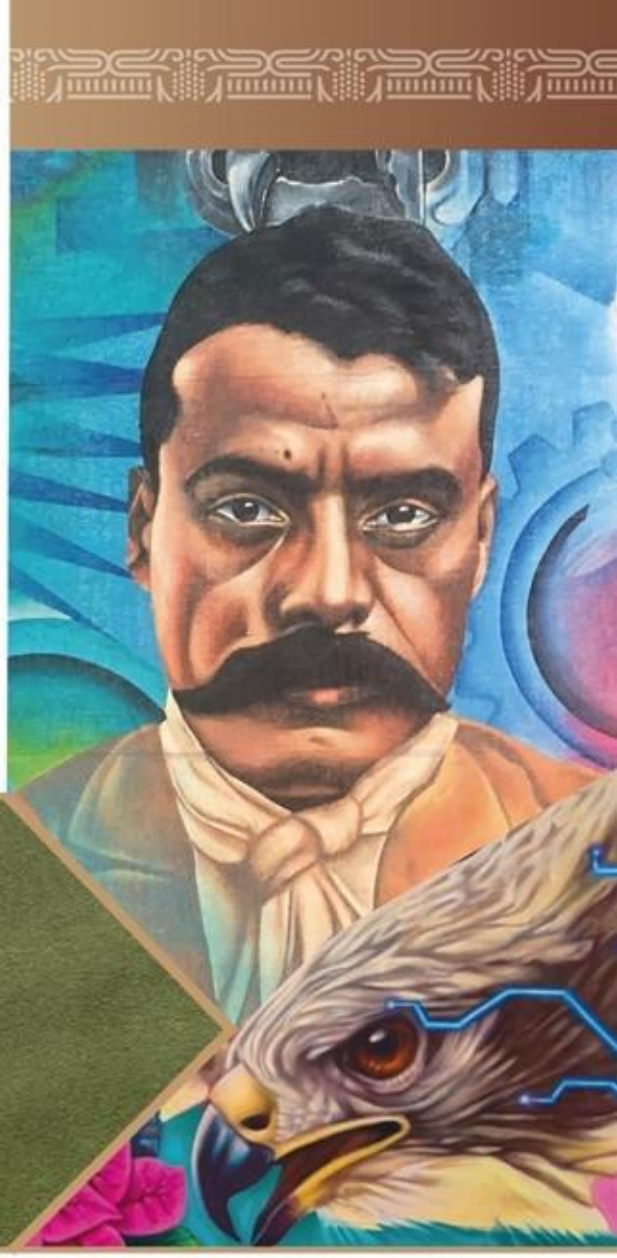
UNIVERSIDAD TECNOLÓGICA EMILIANO ZAPATA DEL ESTADO DE MORELOS

DIVISIÓN ACADÉMICA DE TECNOLOGÍAS DE LA INFORMACIÓN Y DISEÑO

REPORTE DE INTEGRADORA



DESARROLLO DE SOFTWARE MULTIPLATAFORMA

APLICACIONES DE IOT

5°B

EQUIPO:
AVILES SOTELO CHRISTIAN JESÚS
NUÑEZ LUCENA JOSÉ ANGEL
PEDRAZA LÓPEZ OSCAR GIOVANNI
TORRES MÉNDEZ MILCA
VILLALOBOS HERNÁNDEZ DIANA

UTEZ
UNIVERSIDAD TECNOLÓGICA EMILIANO ZAPATA DEL ESTADO DE MORELOS

División Académica de Tecnologías de la Información y Diseño

EMILIANO ZAPATA, MOR., 21 DE ABRIL DE 2025

Índice

Resumen

El objetivo principal de este trabajo fue el desarrollo de un sistema automatizado capaz de monitorear los niveles de dióxido de carbono (CO_2) en un ambiente cerrado y accionar mecanismos de ventilación cuando dichos niveles superaban un umbral previamente establecido, de acuerdo con los valores recomendados para mantener una buena calidad del aire para el ser humano.

Para la ejecución del proyecto, se empleó una metodología centrada en la creación de un prototipo funcional que integró sensores de CO_2, una pantalla OLED para la visualización de datos, LEDs, un buzzer, un sistema de apertura automática de ventilas mediante un servomotor y conectividad WiFi. Esta última permitió la supervisión remota de los niveles de CO_2 a través de una plataforma IoT (Blynk IoT) y un dispositivo móvil, donde se presentaban los datos. Asimismo, se implementó una función de control manual mediante un botón en el Dashboard, lo que permitió al usuario cerrar las ventilas de manera voluntaria.

Entre los resultados más relevantes, se destacó el correcto funcionamiento del sistema de monitoreo, que activaba la apertura automática de las ventilas al superar los niveles de CO_2 el límite establecido. El LED verde indicaba cuando los valores permanecían dentro del rango saludable, mientras que el LED rojo, acompañado del buzzer, alertaba sobre valores elevados. La integración con la plataforma IoT facilitó el monitoreo en tiempo real y brindó al usuario control remoto sobre el sistema.

En conclusión, el sistema desarrollado cumplió con los objetivos propuestos, ofreciendo una solución eficaz para mejorar la calidad del aire en espacios cerrados.

Marco teórico

Estructura IoT

Protocolo de comunicación

El sistema utilizó el protocolo HTTP (Hypertext Transfer Protocol) para el envío de datos desde el dispositivo al servidor de la plataforma IoT. HTTP es un protocolo ampliamente utilizado en internet y también es compatible con aplicaciones IoT, especialmente aquellas que requieren integración con servicios web o APIs.

A través de este protocolo, el microcontrolador realizó solicitudes HTTP tipo POST para enviar los valores de CO_2 capturados hacia la plataforma IoT, donde fueron almacenados y posteriormente visualizados mediante gráficos y paneles de control. Aunque HTTP no es tan ligero como otros protocolos como MQTT, su implementación fue sencilla y efectiva, permitiendo una transmisión confiable de los datos y una integración directa con plataformas basadas en la web.

Normas de calidad del aire

Categorías según niveles de CO2

La categorización de la calidad del aire interior se basa en las concentraciones de CO2, que se miden en partes por millón (ppm):

Hasta 350 ppm: Calidad de aire interior alta.

Entre 350 y 500 ppm: Calidad de aire interior buena.

Entre 500 y 800 ppm: Calidad de aire interior moderada.

Entre 800 y 1200 ppm: Calidad de aire interior baja.

Superior a 1200 ppm: Calidad de aire interior mala.

Estos niveles no solo sirven como referencia para la salud y confort de los ocupantes, sino también para la gestión y diseño de sistemas de ventilación en edificios.

Plataforma IoT utilizada

Se utilizó la plataforma Blynk que permitió:

- Recibir datos del sensor en tiempo real.
- Visualizar gráficos de series temporales.
- Controlar las ventilas con un botón virtual desde un smartphone.
- Configurar alertas visuales (como widgets de notificación o LEDs virtuales).

Software y lenguaje de programación utilizado.

El sistema fue desarrollado utilizando el entorno de programación Arduino IDE con el microcontrolador ESP8266 (o ESP32). El lenguaje de programación utilizado fue C++, adaptado al entorno de Arduino.

Se incluyeron librerías para:

- Lectura del sensor de CO_2.
- Control de servomotores.
- Comunicación WiFi.
- Envío de datos a la plataforma IoT mediante HTTP.
- Control de la pantalla OLED y dispositivos de salida (LEDs y buzzer).
- Configurar alertas visuales (como widgets de notificación o LEDs virtuales).

Objetivos

General

Desarrollar un sistema IoT capaz de monitorear los niveles de CO_2 en espacios cerrados y activar mecanismos de ventilación automáticos o manuales, con visualización en tiempo real tanto física como remota, para mejorar la calidad del aire y prevenir riesgos para la salud.

Específicos

Diseñar un sistema de monitoreo que detectara el aumento en los niveles de CO_2 simulados y mostrara en tiempo real los valores en una pantalla OLED integrada.

Programar el sistema para que active automáticamente un mecanismo de ventilación (servomotor) y alertas cuando los niveles de CO_2 excedan el umbral recomendado.

Configurar una plataforma IoT que reciba, almacene y visualice los datos de CO_2 en formato de series temporales, permitiendo también el control remoto de las ventilas.

Materiales y Metodología

Materiales

- Microcontrolador ESP8266
- Sensor de CO_2 Cjmcu-811
- Pantalla OLED

- Servomotor
- LED verde
- LED rojo
- Buzzer
- Protoboard
- Jumpers
- Cable de datos
- Plataforma IoT Blynk
- Computadora con Arduino IDE

Caracterización de los sensores

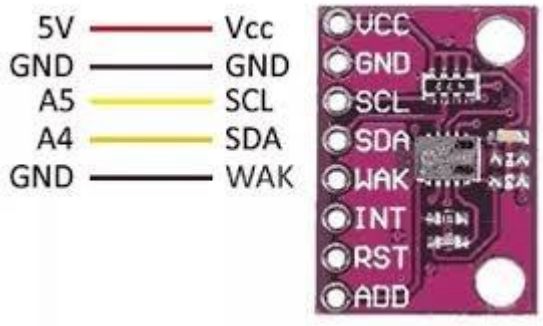
El sensor CJMCU-811 fue probado para asegurar su correcto funcionamiento antes de integrarlo al sistema, este sensor mide niveles de CO_2 .

- Se verificó que sus lecturas se encontraran dentro del rango esperado: 400 a 8192 ppm para CO_2.
- Se realizaron pruebas en un ambiente ventilado y los valores iniciales fueron cercanos a 400 ppm, lo cual indicó una correcta calibración.
- Se expuso al aliento humano y se observó un aumento gradual en las lecturas de CO_2, confirmando su sensibilidad.

Procedimiento

Para el desarrollo del prototipo, se utilizó un sensor de calidad del aire, el cual fue conectado al microcontrolador ESP8266.

Se identificaron los pines del sensor según el diagrama de conexión:



Colocación de los componentes

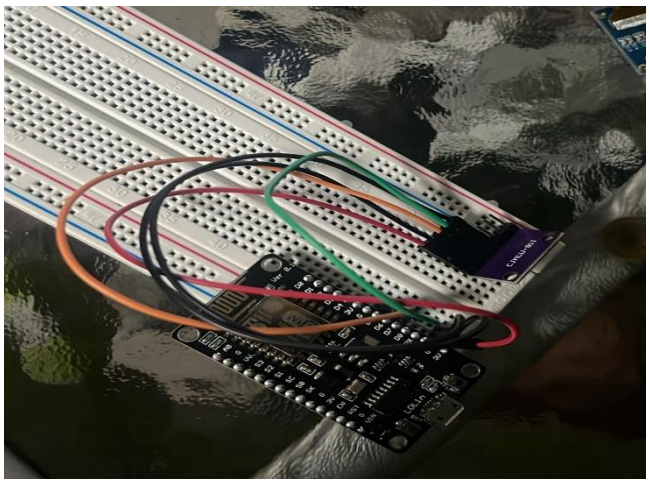
Se colocaron el ESP8266 NodeMCU y el sensor de ca lidad del aire sobre una protoboard, asegurando que los pines quedaran accesibles para facilitar las conexiones.

Identificamos los pines

Se identificaron los pines del sensor como sigue: VCC: alimentación, GND: tierra SCL: línea de reloj del protocolo I2C, SDA: línea de datos del protocolo I2C, WAK: pin de activación del sensor.

Realización de las conexiones

Se conectaron los cables desde el sensor al ESP8266 respetando el siguiente orden: VCC del sensor al pin 3V del ESP8266 (cable rojo) ,GND del sensor al pin GND del ESP8266 (cable negro), SCL del sensor al pin D1 (GPIO5) del ESP8266 (cable verde), SDA del sensor al pin D2 (GPIO4) del ESP8266 (cable naranja), WAK del sensor a GND, manteniéndolo en bajo para que el sensor permaneciera activo.



Verificación del circuito

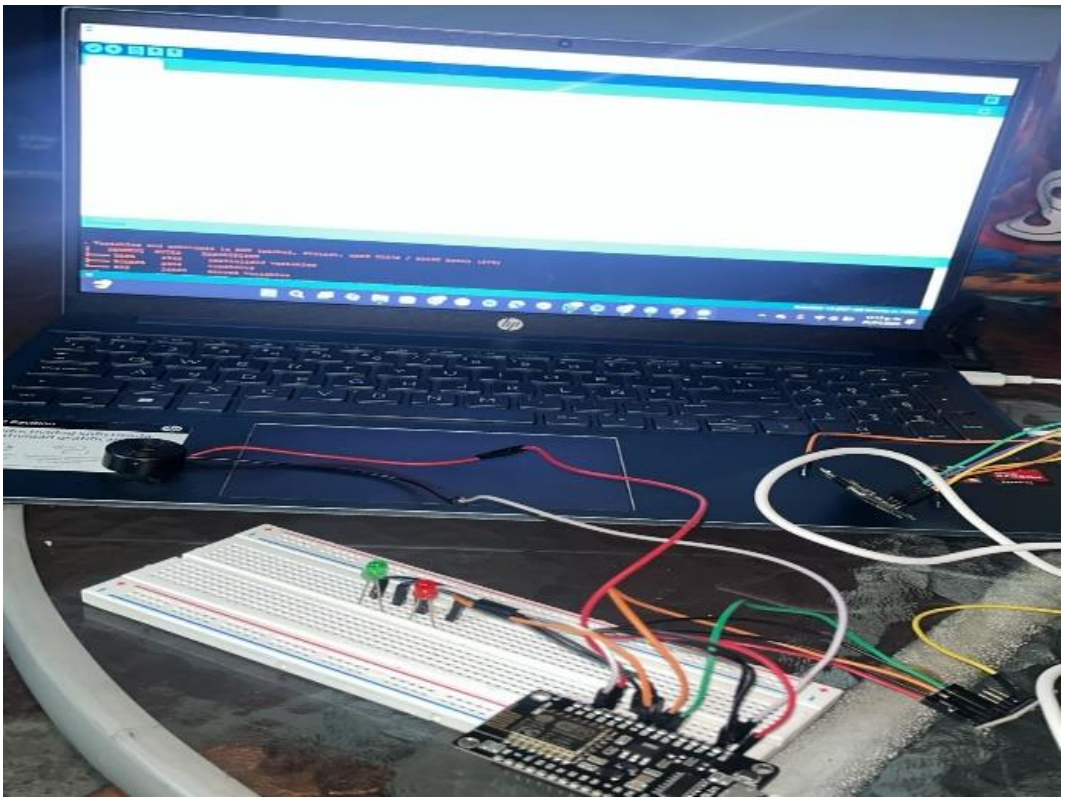
Antes de cargar el programa al ESP8266, se revisaron todas las conexiones del circuito montado sobre la protoboard. Se comprobó que cada componente estuviera correctamente conectado, siguiendo el esquema previsto.

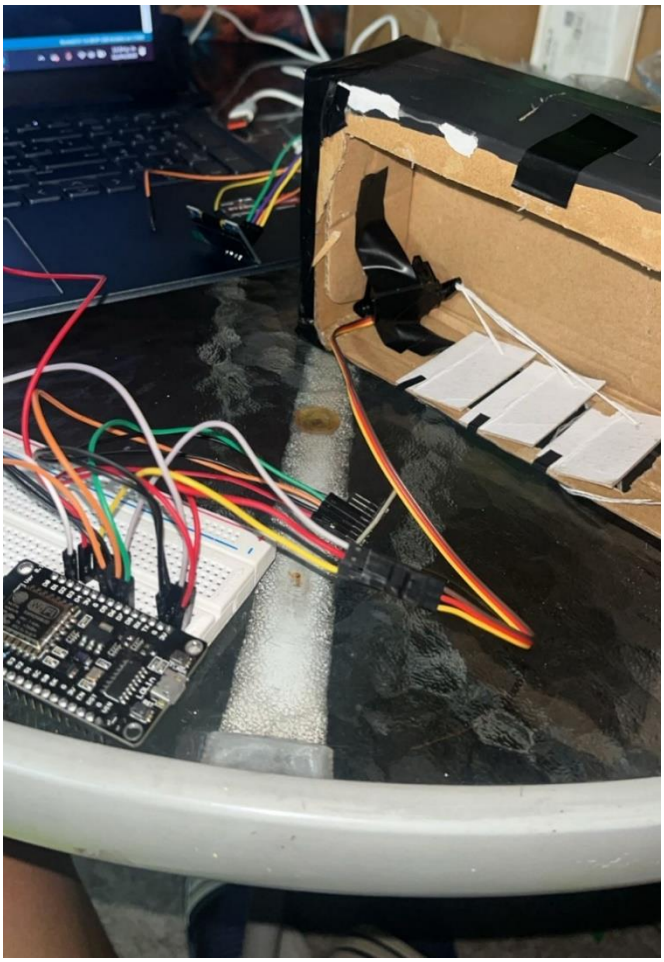
Primero, se verificó que los LEDs estuvieran conectados correctamente: el ánodo (pierna larga) a las salidas digitales del ESP8266 y el cátodo (pierna corta) a GND por medio de una resistencia. Luego, se comprobó que el buzzer tuviera su pin positivo conectado al pin correspondiente del ESP8266 y el negativo a tierra. También se revisó el cableado del servomotor, asegurándose de que los cables de VCC, GND y señal estuvieran bien conectados a los pines del microcontrolador.

Una vez verificado todo, se conectó el ESP8266 a la computadora usando un cable micro USB. Esto permitió alimentar el sistema y cargar el programa desde el entorno de desarrollo Arduino IDE. Se seleccionó el puerto COM adecuado y se compiló el código sin errores.

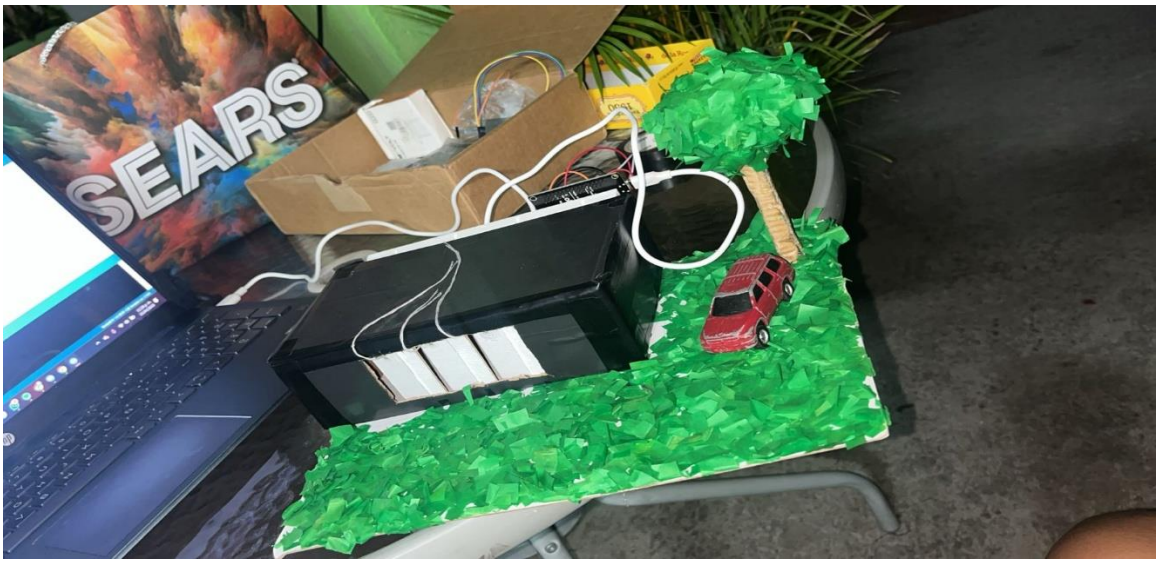
Después de cargar el programa, se observó el funcionamiento del circuito: los LEDs encendieron según la lógica programada, el buzzer emitió un sonido y el servomotor se movió correctamente. Esto confirmó que tanto el código como el cableado estaban bien.

Por último, se tuvo en cuenta que, si se conectan más componentes en el futuro, puede ser necesario usar una fuente de alimentación externa para evitar sobrecargar el ESP8266.

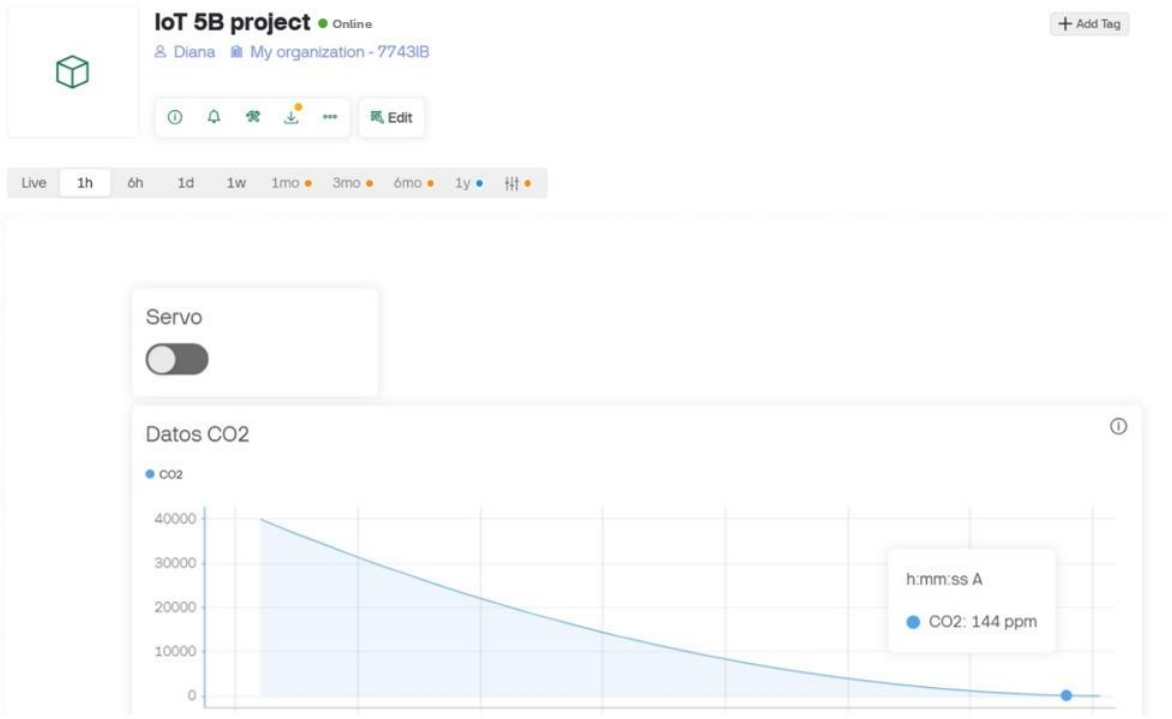


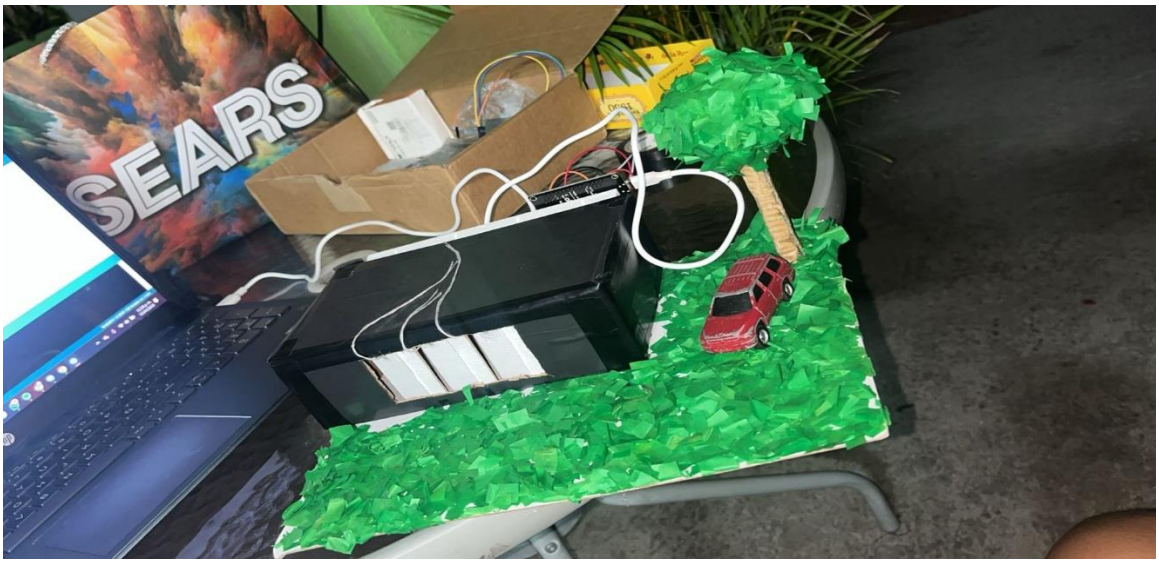


Se realizo un maquetado de una casa con patio en el cual simulamos una casa para ocultar el circuito y se visualizara más estético.


SEARS

Resultados


IoT 5B project
Online
Diana
My organization - 7743IB
Add Tag
Edit
Live
1h
6h
1d
1w
1mo
3mo
6mo
1y
Servo
Datos CO2
co2
40000
30000
20000
10000
0
h:mm:ss A
CO2: 144 ppm



Conclusiones

De acuerdo a los resultados obtenidos durante el desarrollo y prueba del sistema, se concluyó que los objetivos planteados fueron cumplidos de manera satisfactoria.

El objetivo general se alcanzó, ya que se logró desarrollar un sistema IoT funcional capaz de monitorear los niveles de CO_2 en tiempo real. Además, el sistema activó automáticamente un mecanismo de ventilación y alertas cuando los valores superaron el umbral establecido, ayudando a mejorar la calidad del aire del entorno simulado.

Se diseñó y se implementó correctamente el sistema de monitoreo, el cual detectó con precisión los aumentos en los niveles de CO_2 simulados.

Se programó exitosamente el control para activar el servomotor y los indicadores cuando el sensor detectó niveles elevados de CO_2, respondiendo de manera automática.

Se configuró y se utilizó una plataforma IoT (Blynk) que permitió visualizar los datos como series temporales en la app móvil y controlar manualmente la apertura o cierre de la ventila, cumpliendo con los requisitos de monitoreo y control remoto.

Por lo tanto, el proyecto funcionó según lo esperado y demostró ser una solución viable para espacios cerrados donde se requiera mantener una buena calidad del aire.

Bibliografías

S&Amp;P. (2024, 9 diciembre). Efectos del CO2 en la calidad del aire interior y la salud | S&P. S&P Sistemas de Ventilación. https://www.solerpalau.com/es-es/blog/efectos-co2/

Anexos

Código del programa

```
// Definición de parámetros de Blynk (sin punto y coma al final)
#define BLYNK_TEMPLATE_ID      "TMPL2rMqGMTxd"       // ID de la plantilla en Blynk
#define BLYNK_TEMPLATE_NAME   "IoT 5B project"     // Nombre de la plantilla
#define BLYNK_AUTH_TOKEN       "wddAULjWDkSGu4jbM8RiLg7_hYCjRw4k"  // Token de autenticación

// Librerías necesarias
#include <Wire.h>                  // Comunicación I2C
#include <Adafruit_CCS811.h>       // Sensor de CO2 y TVOC CCS811
#include <Adafruit_GFX.h>          // Core de gráficos de Adafruit
#include <Adafruit_SSD1306.h>      // Pantalla OLED SSD1306
#include <Servo.h>                 // Control de servomotor
#include <ESP8266WiFi.h>           // Conexión WiFi en ESP8266
#include <BlynkSimpleEsp8266.h>    // Cliente Blynk para ESP8266

// Pines definidos en la placa (NodeMCU/WeMos D1 Mini)
#define SDA       D6    // Línea SDA para I2C
#define SCL       D5    // Línea SCL para I2C
#define LEDROJO   D1    // LED rojo indicador de alarma
#define LEDVERDE  D4    // LED verde indicador de estado normal
#define BUZZER    D2    // Buzzer para alerta sonora
#define SERVO     D3    // Pin de control del servomotor

// Credenciales de tu red WiFi
char ssid[] = "Totalplay-0FA5-5G";
char pass[] = "0FA5073Aq9W5Fu3N";

// Objetos globales
Adafruit_CCS811 ccs;                    // Sensor de gas CCS811
```

```
Adafruit_SSD1306 display(128, 64, &Wire, -1);     // Pantalla OLED 128×64
Servo servo;                                      // Servomotor

// Variables de estado
bool alarmaActiva = false;    // ¿La alarma está sonando?
bool controlManual = false;   // ¿Está el servo en modo manual?

void setup() {
  // Inicialización serie para debug
  Serial.begin(115200);
  delay(1000);

  // Arranque del bus I2C con los pines definidos
  Wire.begin(SDA, SCL);

  // Configuración de pines de salida
  pinMode(LEDROJO,   OUTPUT);
  pinMode(LEDVERDE,  OUTPUT);
  pinMode(BUZZER,    OUTPUT);
  // Apagar indicadores al inicio
  digitalWrite(LEDROJO,  LOW);
  digitalWrite(LEDVERDE, LOW);
  digitalWrite(BUZZER,   LOW);

  // Configuración e inicialización del servomotor
  servo.attach(SERVO);
  servo.write(180);  // Posición “cerrada” al inicio

  // Inicializar pantalla OLED
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println("No se encontró la pantalla OLED");
    while (true);  // Si falla, quedamos en bucle infinito
  }
  // Mensaje inicial en pantalla
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println("Iniciando sensor de CO2...");
  display.display();
```

```
  // Inicializar sensor CCS811
  if (!ccs.begin()) {
    display.clearDisplay();
    display.setCursor(0, 0);
    display.println("Error al iniciar sensor de CO2");
    display.display();
    while (true);  // Bucle infinito si hay error
  }

  // Conectar a Blynk
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

  // Esperar hasta que el sensor esté listo o se agote el tiempo
  unsigned long start = millis();
  while (!ccs.available()) {
    if (millis() - start > 5000) break;  // 5 s de espera máxima
    delay(100);
  }

  // Mensaje de medición en pantalla
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("Midiendo niveles de CO2...");
  display.display();
  delay(1000);
}

// Handler para el widget en V3 de Blynk (control manual del servo)
BLYNK_WRITE(V3) {
  int estado = param.asInt();  // 0 o 1 desde Blynk
  controlManual = estado;
  if (controlManual) {
    servo.write(0);  // Abrir servomotor manualmente
  } else if (!alarmaActiva) {
    servo.write(0);  // Reiniciar posición si no hay alarma
  }
}

void loop() {
```

```
Blynk.run();  // Mantenimiento de la conexión Blynk

// Lectura del sensor; si hay error mostramos alerta en pantalla
if (!ccs.available() || ccs.readData()) {
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("¡Hubo un error!");
  display.display();
  return;
}

// Obtener ppm de CO2
uint16_t co2 = ccs.geteCO2();

// Mostrar valor en pantalla
display.clearDisplay();
display.setCursor(0, 0);
display.print("CO2: ");
display.print(co2);
display.println(" ppm");
display.display();

// Enviar dato a Blynk en V2
Blynk.virtualWrite(V2, co2);

// Lógica de alarma y control de salidas
if (co2 < 800) {
  // Estado normal
  if (!controlManual) {
    servo.write(180);  // Cerrar ventilación
  }
  digitalWrite(LEDVERDE, HIGH);
  digitalWrite(LEDROJO,  LOW);
  digitalWrite(BUZZER,   LOW);
  alarmaActiva = false;
} else {
  // Alarma por CO2 alto
  digitalWrite(LEDROJO,  HIGH);
  digitalWrite(LEDVERDE, LOW);
  digitalWrite(BUZZER,   HIGH);
```

```
    servo.write(0);       // Abrir ventilación para ventilar
    alarmaActiva = true;
  }
  delay(2000);  // Pausa de 2 segundos entre lecturas
}
```

Diagrama de conexión