**Instituto Tecnológico y de Estudios Superiores de Monterrey**

**Campus Ciudad de México**


**Implementación del Internet en las cosas**

**Nombre del profesor:**

*Octavio Navarro Hinojosa*
*Ruben Raya Delgado*
*Martha Sara De Gante Velazquez*

# Situación Problema

Equipo 1 | Integrantes:

| | | |
|---|---|---|
| Alejandro Ostos Roji | IID | A01552398 |
| Thomas Garcia Estebarena | IID | A01662503 |
| Iñigo Orozco Rodriguez | IID | A01783125 |
| Jorge Siegrist | IID | A01782873 |
| Francisco Javier Romero Davy | IID | A01783170 |

16 oct 2023

**Contenido:**

## Resumen del Proyecto:

El proyecto tiene como objetivo crear un sistema de monitoreo ambiental que recopile y visualice datos de temperatura, humedad y calidad del aire utilizando tres sensores (DHT, MQ135 y RTC 1302). Los datos son procesados por una placa Arduino y enviados a un servidor web mediante una conexión WiFi. El servidor web, creado con Flask, recibe los datos y los almacena en una base de datos MySQL. Luego, se utiliza un programa Python junto con Dash para visualizar los datos en forma de gráficas interactivas.

## Introducción:

Durante estas diez semanas, trabajamos en un proyecto el cual consiste en diseñar y construir un prototipo de sistema digital que nos permitirá recopilar y analizar datos como gas, temperatura y tiempo. Este sistema será capaz de obtener información a través de tres sensores fundamentales: DHT11, MQ135 y RTC 1302.

Para llevar a cabo este proyecto, utilizaremos la plataforma de desarrollo Arduino, Github y MySQL, las cuales nos brindan la flexibilidad y las herramientas necesarias para programar y conectar nuestros sensores de manera eficiente.

Los códigos proporcionados, los cuales están a lo largo de este reporte, son la base de nuestro proyecto. Este código controla la interacción entre los sensores, la pantalla LCD y la conexión a la plataforma en línea donde depositamos la información recolectada para su posterior análisis y visualización.

## Materiales:

| Componente | Función |
|---|---|

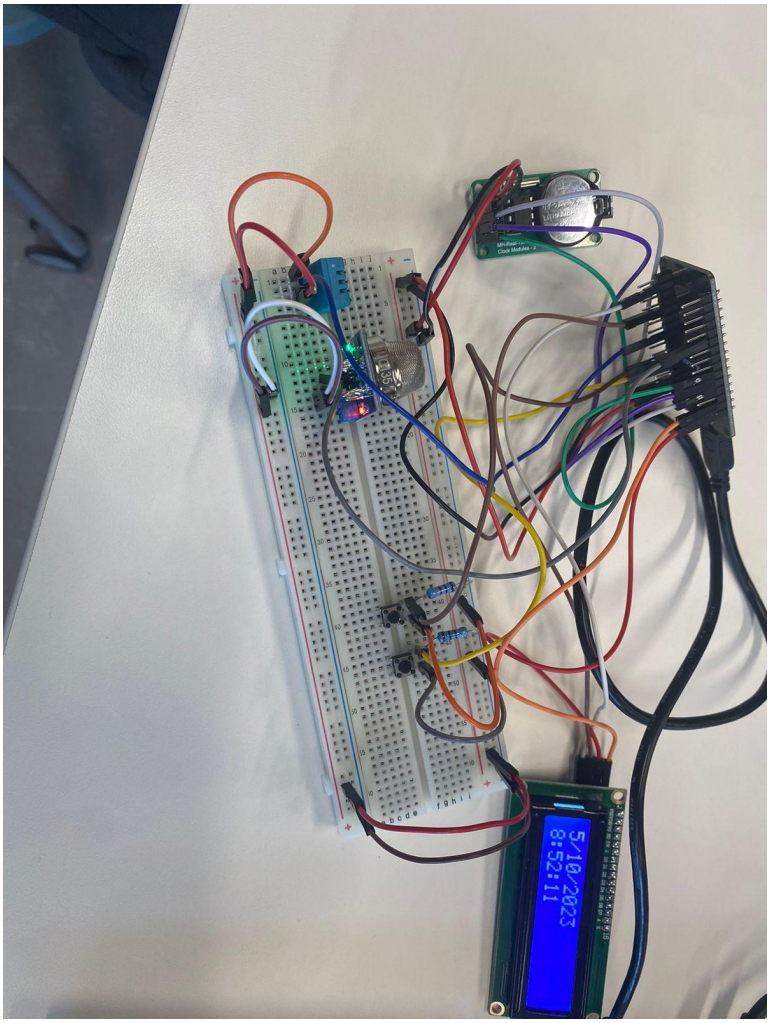
| | |
|---|---|
| **Sensor de gas MQ135** | Sensor de gases que mide la calidad del aire detectando concentraciones de dióxido de carbono y otros elementos perjudiciales. |
| **Sensor de temperatura y humedad DHT11** | Sensor combinado de temperatura y humedad que proporciona lecturas precisas de ambos parámetros. |
| **Sensor de tiempo RTC 1302** | Sensor de tiempo en tiempo real que asegura un registro preciso de la fecha y la hora. |
| **Display LCD** | Muestra información como fecha, hora y datos de sensores. Facilita la visualización y comprensión de los datos en tiempo real. |
| **Protoboard** | Plataforma de pruebas que permite la conexión y prueba de múltiples componentes electrónicos. |
| **ESP32** | Microcontrolador que gestiona la comunicación WiFi y la transferencia de datos entre los sensores y el servidor. |
| **Push Buttons** | Botones de control utilizados para cambiar entre modos de visualización en la pantalla LCD. |
| **Jumpers** | Cables conductores utilizados para establecer conexiones entre los diferentes componentes, permitiendo el flujo de datos y la energía a través del circuito. |

***(Tabla 1)***

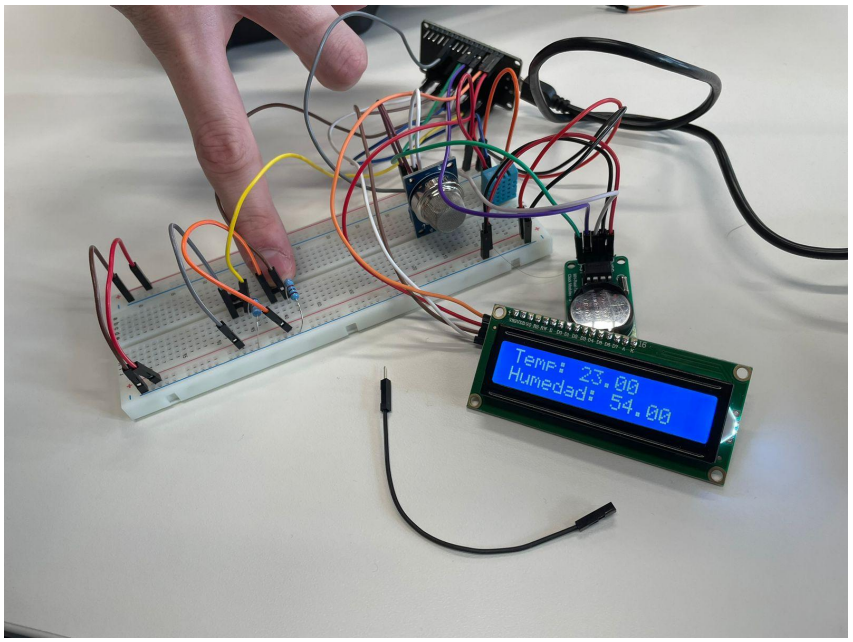
## Desarrollo del Proyecto:

### Configuración de Componentes

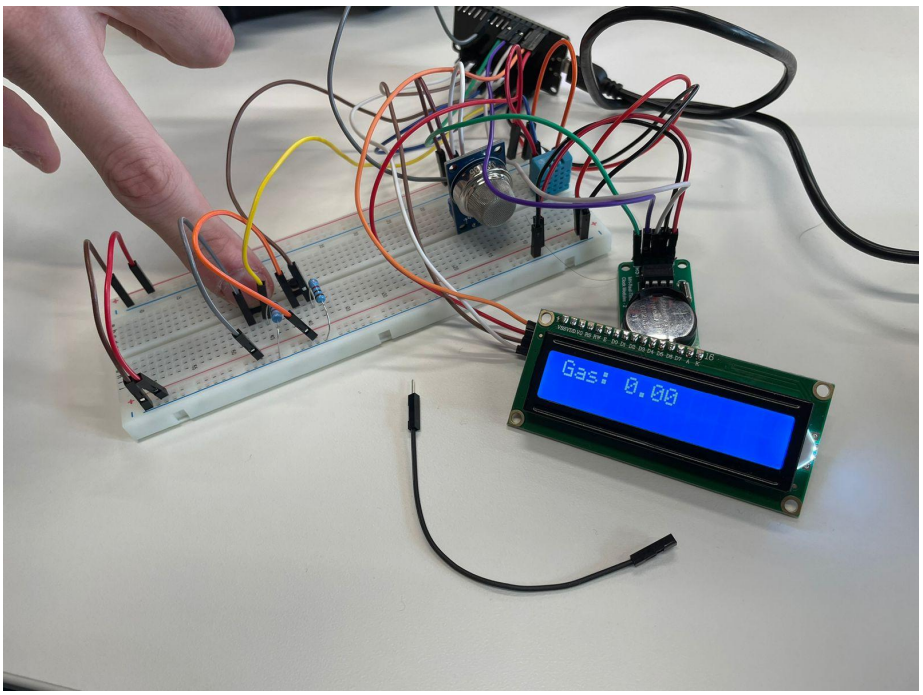
Se utilizó un Arduino Uno como la unidad principal de control. Se conectaron tres sensores: DHT para medir la temperatura y humedad, MQ135 para la calidad del aire y RTC 1302 para mantener un registro preciso de la fecha y hora. Estos sensores fueron dispuestos en una protoboard junto con una pantalla LCD I2C y dos botones de control.




*(Ilustración Prototipo funcional. RTC 1302)*




*(Ilustración 2 Prototipo funcional. Medición de DHT11)*




*(Ilustración 3 Prototipo funcional. Medición MQ135)*

## Programación del Arduino

El código Arduino se encarga de varias tareas cruciales:

1. **Inicialización de Componentes:** Se incluyeron las librerías necesarias para comunicarse con los diferentes componentes, como el sensor RTC, el sensor DHT, la pantalla LCD y el sensor MQ135.

2. **Configuración de Conexión WiFi:** Se estableció la conexión a una red WiFi, proporcionando las credenciales necesarias.

3. **Lectura de Sensores y Modos de Visualización:** El bucle principal del programa realiza la lectura de los sensores DHT y MQ135, así como del reloj RTC. Además, se implementa un sistema de modos de visualización controlado por los botones. Esto permite cambiar entre la visualización de fecha y hora, temperatura/humedad y calidad del aire.

4. **Envío de Datos al Servidor:** Siempre que se realiza una lectura, los datos se envían al servidor a través de una solicitud HTTP POST. Esto asegura que los datos se actualicen en tiempo real en la aplicación web.

## Servidor Flask y Base de Datos MySQL

El servidor Flask actúa como el intermediario entre el Arduino y la base de datos. Las funciones clave del servidor son:

1. **Conexión a la Base de Datos:** Se implementó una función para conectar el servidor a la base de datos MySQL. Si la conexión falla, se proporciona un mensaje de error.

2. **Recepción de Datos:** La ruta (/sensor_data) está configurada para recibir datos de Arduino a través de una solicitud POST. Los datos se extraen del cuerpo de la solicitud y se preparan para su inserción en la base de datos.

3. **Inserción en la Base de Datos:** Los datos se insertan en la base de datos utilizando una consulta SQL preparada. Si la inserción es exitosa, se confirma.

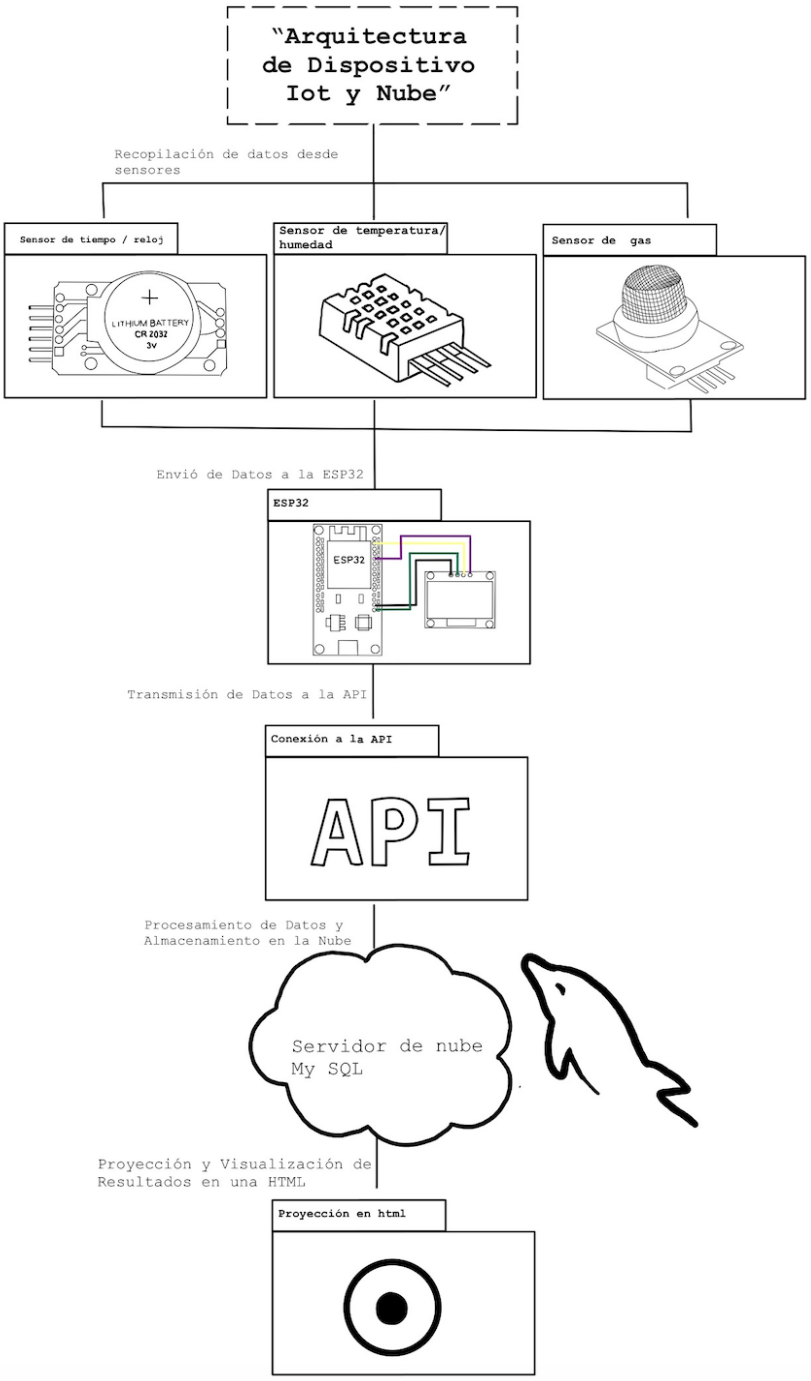
### Visualización de Datos con Dash

El programa Python utiliza Dash, una librería de Python para crear aplicaciones web interactivas, para visualizar los datos almacenados. Se siguen estos pasos:

1. **Conexión a la Base de Datos:** Se establece una conexión a la base de datos MySQL para recuperar los datos almacenados.

2. **Creación de Gráficas Interactivas:** Utilizando la librería Plotly, se crean gráficas de humedad y temperatura en función del tiempo. Estas gráficas permiten al usuario explorar los datos de manera interactiva.
   Interfaz de Usuario Intuitiva: Se crea una interfaz de usuario que muestra las gráficas y proporciona una breve descripción del propósito de la aplicación.

## Arquitectura del sistema:




**(Ilustración 4)**

**Paso 1: Recopilación de Datos desde Sensores**

La arquitectura comienza con la recopilación de datos de tres sensores diferentes: un sensor de temperatura y humedad (DHT11), un sensor de calidad del aire y gases (MQ-135) y un reloj en tiempo real (RTC). Estos sensores están diseñados para capturar información vital sobre las condiciones en un entorno específico.

**Paso 2: Envío de Datos a la ESP32**

Una vez que los sensores han adquirido los datos, se procede al envío de esta información hacia un dispositivo local llamado ESP32. La ESP32 actúa como el nodo central del sistema y es responsable de recibir, procesar y transmitir los datos a lo largo de la arquitectura.

**Paso 3: Transmisión de Datos a la API**

La ESP32 actúa como intermediario y se conecta con una API (Interfaz de Programación de Aplicaciones) a través de una conexión a la red. Esta API puede realizar tareas adicionales de procesamiento y validación de datos antes de que los datos se envíen al siguiente componente.

**Paso 4: Procesamiento de Datos y Almacenamiento en la Nube**

En este paso, la API recibe los datos del dispositivo local y procede a procesarlos. Esto incluye la validación de datos, transformaciones necesarias y cálculos adicionales si es requerido. Una vez que los datos han sido procesados y verificados, se inyectan en la nube.

Los datos procesados se almacenan en una base de datos en la nube, en este caso, se utiliza MySQL como el sistema de gestión de bases de datos. Los datos se organizan y registran en la base de datos, cada uno acompañado de una marca de tiempo para el seguimiento temporal.

**Paso 5: Proyección y Visualización de Resultados en una Aplicación HTML**

Los datos almacenados en la base de datos MySQL están disponibles para su acceso. Una aplicación web o página HTML se utiliza para consultar y recuperar estos datos. La aplicación toma los datos y los proyecta en forma de gráficos, informes o visualizaciones comprensibles para los usuarios finales.

Estas visualizaciones permiten a los usuarios monitorear y comprender fácilmente las condiciones registradas por los sensores en tiempo real o a lo largo del tiempo.

# Código Github (ShowData):

```
import mysql.connector
import plotly.express as px
import pandas as pd
from dash import Dash, html, dcc
from flask import Flask, request, jsonify


def createConnection(user_name, database_name, user_password, host, port):
    cnx = mysql.connector.connect(user=user_name, database=database_name, password=user_password, host=host, port=port)
    cursor = cnx.cursor()
    return (cnx, cursor)

data_count = 0
sensor_data = []

def fetch_data():
    try:
        cnx, cursor = createConnection('sql10652872', ' sql10652872', 'IljS3LDZZm', 'sql10.freemysqlhosting.net', '3306')
        query = "SELECT humidity, temperature, gas_value, date_time FROM dht_sensor_data"
        cursor.execute(query)
        data = cursor.fetchall()
        return data

    except mysql.connector.Error as err:
        print(err)

if __name__ == '__main__':
    data = fetch_data()
    print(data)  # Imprime los datos recuperados de la base de datos
    df = pd.DataFrame(data, columns=["humidity", "temperature", "gas_value", "date_time"])
    print(df)
if __name__ == '__main__':
```

```
data = fetch_data()
df = pd.DataFrame(data, columns=["humidity", "temperature", "gas_value", "date_time"])

# Asegúrate de que los tipos de datos sean numéricos
df['humidity'] = pd.to_numeric(df['humidity'])
df['temperature'] = pd.to_numeric(df['temperature'])
df['gas_value'] = pd.to_numeric(df['gas_value'])

# Convierte 'date_time' a formato de fecha si no está en ese formato
df['date_time'] = pd.to_datetime(df['date_time'])

app = Dash(__name__)
app.layout = html.Div([
    html.Div(
        children=[
            html.H1("IOT Data Visualization", style={'text-align': 'center'}),
            html.P("This graph shows humidity, temperature values over time."),
            dcc.Graph(figure=px.line(df, x='date_time', y=['humidity', 'temperature'], title="Humidity, Temperature vs
            html.P("This graph shows gas values over time"),
            dcc.Graph(figure=px.line(df, x='date_time', y=['gas_value'], title= "Gas vs Time")),
        ])
])

app.run_server(debug=True)
```

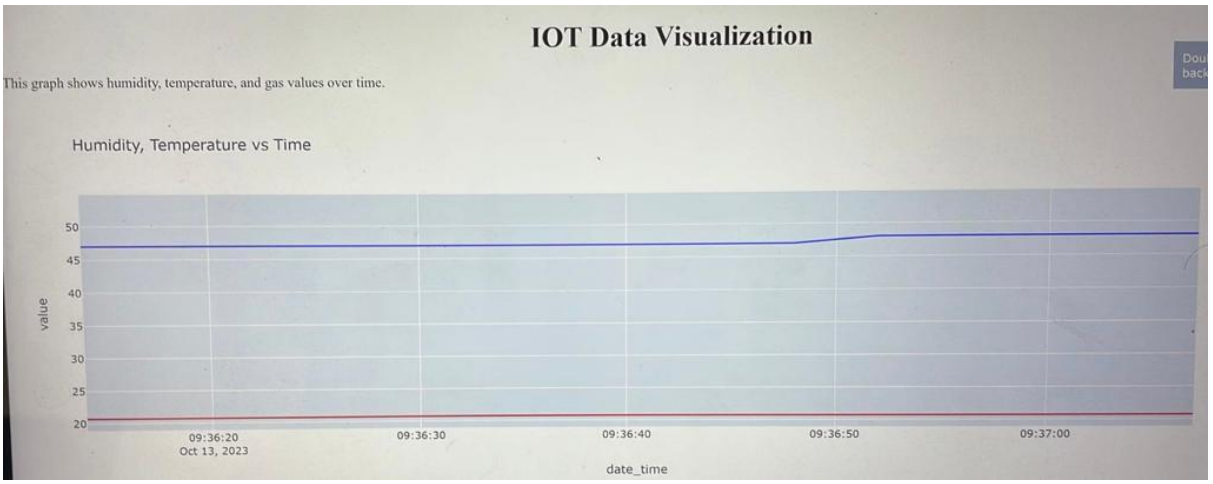
**Descripción:**

Nuestro código show data se encarga de interactuar con una base de datos MySQL para recuperar y presentar los datos del sistema de monitoreo ambiental. Para lograr esto, utiliza diversas librerías como mysql.connector, plotly.express y pandas.

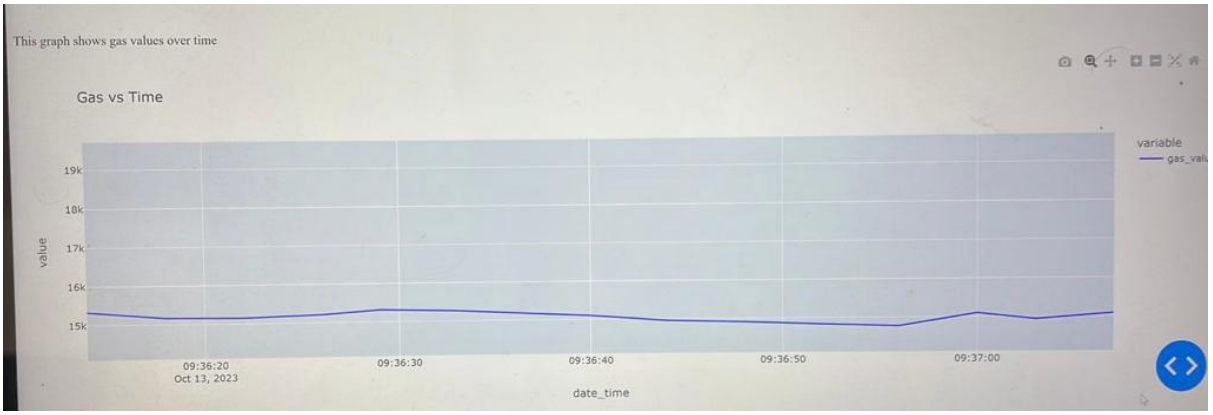
Primero, se importan las librerías necesarias para la conexión y manipulación de datos en la base de datos MySQL. Luego, se definen funciones, como createConnection para establecer la conexión y fetch_data para obtener los datos.

Después de establecer la conexión, se ejecuta una consulta SQL para seleccionar los datos de fecha, humedad y temperatura de nuestra base de datos "dht_sensor_data". Los resultados se almacenan en un DataFrame de Pandas.

Luego, se crea una aplicación web interactiva utilizando Dash. Esto incluye la definición de un título, una descripción y una gráfica interactiva que muestra la evolución de la humedad y temperatura a lo largo del tiempo.




*(Ilustración 5 Visualización de datos en página web. Temperatura y humedad vs Time)*




*(Ilustración 6 Visualización de datos en página web. GasValue vs Time)*

## Código Github (Servidor FLASK)

```python
from flask import Flask, request, jsonify
import mysql.connector

app = Flask(__name__)

# Función para crear la conexión con la base de datos MySQL
def createConnection(user, password, host, database, port):
    try:
        # Intenta establecer una conexión con la base de datos MySQL
        cnx = mysql.connector.connect(user=user, password=password, host=host, database=database, port=port)
        cursor = cnx.cursor()
        return cnx, cursor
    except Exception as e:
        # Si hay algún error, imprime el mensaje de error
        print("Database connection error:", str(e))
        return None, None  # Devuelve valores None en caso de error

# Ruta para recibir datos del sensor a través de una solicitud POST
@app.route('/sensor_data', methods=['POST'])
def receive_data():
    global data_count  # Variable global para contar los datos recibidos

    try:
        data = request.json
        print(data.get('date_time'))  # Imprime la fecha y hora recibidas
        print(data.get('humidity'))  # Imprime la humedad recibida
        print(data.get('temperature'))  # Imprime la temperatura recibida
        print(data.get('gasValue'))  # Imprime el valor del gas recibido
        humidity = float(data.get('humidity'))  # Convierte la humedad a tipo flotante
        temperature = float(data.get('temperature'))  # Convierte la temperatura a tipo flotante
        gas_value = float(data.get('gasValue'))  # Convierte el valor del gas a tipo flotante
        datestring = str(data.get('date_time'))  # Convierte la fecha y hora a tipo cadena

        # Agrega los datos a la lista (comentado, ya que la lista 'sensor_data' no está definida en el código)
        # sensor_data.append((humidity, temperature, gas_value))
        print("Recibí datos")

        # Inserta los datos en la base de datos MySQL
        cnx, cursor = createConnection('sql10652872', 'I1jS3LDZZm', 'sql10.freemysqlhosting.net', 'sql10652872', '3306
        query = "INSERT INTO dht_sensor_data (humidity, temperature, gas_value, date_time) VALUES (%s, %s, %s, %s)"
        data = (humidity, temperature, gas_value, datestring)
        cursor.execute(query, data)
        cnx.commit()
        print("Datos insertados correctamente")

    except Exception as e:
        # Si ocurre un error, imprime el mensaje de error
        print("Error:", str(e))

    return jsonify({'message': 'Data received successfully'})  # Devuelve un mensaje en formato JSON indicando que los
```

## Descripción:

El código del servidor Flask establece una aplicación web que actúa como intermediario entre el sistema de monitoreo y la base de datos MySQL. Su función principal es recibir los datos enviados por el sistema de monitoreo y almacenarlos en la base de datos para su posterior acceso y análisis.

El código comienza importando las librerías necesarias, incluyendo Flask para crear la aplicación web y mysql.connector para la interacción con la base de datos MySQL.

El servidor Flask tiene una ruta definida, /sensor_data, que espera recibir datos a través de solicitudes POST. Cuando se recibe una solicitud en esta ruta, la función receive_data es invocada.

Finalmente, se confirman los cambios realizados en la base de datos y se devuelve un mensaje indicando que los datos se han recibido correctamente.

## Código Arduino:

```
#include <Wire.h>
#include <RtcDS1302.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#include <ArduinoJson.h>
#include <HTTPClient.h>
#include <WiFi.h>

ThreeWire myWire(14, 12, 27);
RtcDS1302<ThreeWire> Rtc(myWire);
LiquidCrystal_I2C lcd(0x27, 16, 2);
DHT dht(5, DHT11);
const int mq135Pin = 34;
const char* ssid = "Arris GE";
const char* password = "F0AF85E94DB4";
const int buttonPin1 = 2;
const int buttonPin2 = 4;
const char* apiEndpoint = "https://fictional-space-spoon-979j4x67jrq43pp65-5000.app.github.dev/sensor_data";

enum DisplayMode {
  SHOW_TIME,
  SHOW_DHT,
  SHOW_MQ135
};

DisplayMode currentMode = SHOW_TIME;
float temperature;
float humidity;
float gasValue;


void setupWifi() {
  Serial.begin(9600);
```

```
  Serial.print("Connecting to WiFi");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(100);
    Serial.print(".");
  }
  Serial.print(" Connected: ");
  Serial.println(WiFi.localIP());
}

void setup() {
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();
  lcd.clear();
  currentMode = SHOW_TIME;
  setupWifi();
  pinMode(buttonPin1, INPUT_PULLUP);
  pinMode(buttonPin2, INPUT_PULLUP);

  dht.begin();
  Rtc.Begin();

  RtcDateTime compiled = RtcDateTime(_DATE, __TIME_);
  if (!Rtc.IsDateTimeValid()) {
    Serial.println("RTC no válido. Configurando la fecha y hora...");
    Rtc.SetDateTime(compiled);
  }
}

void loop() {
  RtcDateTime now = Rtc.GetDateTime();
  lcd.clear();
  temperature = dht.readTemperature();
  humidity = dht.readHumidity();
  gasValue = readMQ135();
  sendData(temperature, humidity, gasValue, now);
  if (digitalRead(buttonPin1) == HIGH) {
    currentMode = SHOW_DHT;
  } else if (digitalRead(buttonPin2) == HIGH) {
    currentMode = SHOW_MQ135;
  } else {
    currentMode = SHOW_TIME;
  }

  switch (currentMode) {
    case SHOW_TIME:
      lcd.print(now.Day(), DEC);
      lcd.print('/');
      lcd.print(now.Month(), DEC);
      lcd.print('/');
      lcd.print(now.Year(), DEC);
      lcd.setCursor(0, 1);
      lcd.print(now.Hour(), DEC);
      lcd.print(':');
      lcd.print(now.Minute(), DEC);
      lcd.print(':');
      lcd.print(now.Second(), DEC);
      break;

    case SHOW_DHT:
      temperature = dht.readTemperature();
      humidity = dht.readHumidity();
```

```
      lcd.print("Temp: ");
      lcd.print(temperature);
      lcd.setCursor(0, 1);
      lcd.print("Humedad: ");
      lcd.print(humidity);
      break;

    case SHOW_MQ135:
      gasValue = readMQ135();
      displayMQ135Value();
      break;
  }

  delay(1000);
}

void sendData(float temperature, float humidity, float gasValue, const RtcDateTime& timestamp) {
 Serial.print("Sending data to API: ");

  Serial.print(temperature);
  Serial.print(humidity);
  Serial.print(gasValue);

  HTTPClient http;
  http.begin(apiEndpoint);
  http.addHeader("Content-Type", "application/json");

  StaticJsonDocument <200> doc; // Use DynamicJsonDocument
  doc["temperature"] = temperature;
  doc["humidity"] = humidity;
  doc["gasValue"] = gasValue;

  char datestring[20];
```

```
  snprintf_P(datestring, sizeof(datestring), PSTR("%04u-%02u-%02u %02u:%02u:%02u"),
             timestamp.Year(), timestamp.Month(), timestamp.Day(),
             timestamp.Hour(), timestamp.Minute(), timestamp.Second());
  doc["date_time"] = datestring;

  String json;
  serializeJson(doc, json);

  int httpResponseCode = http.POST(json);
  if (httpResponseCode > 0) {
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
    String responseString = http.getString();
    Serial.println("Received response: " + responseString);
  } else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }
  http.end();
}

float readMQ135() {
  // Leer la lectura analógica del nuevo pin del sensor MQ135
  int sensorValue = analogRead(mq135Pin);

  float gasValue = (float)sensorValue / 1024.0 * 10000.0;

  return gasValue;
}
void displayMQ135Value() {
  float gasValue = readMQ135();
  lcd.print("Gas: ");
  lcd.print(gasValue);
```

**Descripción:**

El código de Arduino coordina la lectura de tres sensores (DHT, MQ135 y RTC1302), controla una pantalla LCD y gestiona dos botones para cambiar entre modos de visualización. También envía los datos a una API remota a través de WiFi, permitiendo el monitoreo de condiciones ambientales.

## Resultados:

El proyecto se desarrolló exitosamente, logrando una integración efectiva entre los componentes. Los sensores DHT, MQ135 y RTC 1302 demostraron una precisión medición de temperatura, humedad y calidad del aire, respectivamente.

La comunicación entre el Arduino y el servidor Flask funcionó de manera fiable a través de la conexión WiFi. Esto permitió una transmisión fluida de datos, que se almacenan de manera correcta en la base de datos MySQL.

La aplicación web creada con Dash proporcionó una interfaz intuitiva y fácil de usar. Las gráficas interactivas permiten una visualización detallada de las condiciones ambientales en tiempo real.

## Conclusión general:

El proyecto ha logrado desarrollar un sistema de monitoreo ambiental completo, desde la adquisición de datos hasta la visualización en una aplicación web. El uso de componentes accesibles y tecnologías de código abierto ha demostrado ser efectivo y escalable. La colaboración entre Arduino y Flask para la transferencia de datos y la administración de la base de datos ha sido un enfoque efectivo.

El sistema ha demostrado su capacidad para proporcionar información en tiempo real sobre la temperatura, humedad y calidad del aire, lo cual es fundamental para el monitoreo ambiental y puede ser útil en una variedad de aplicaciones.

Este proyecto sienta las bases para futuras expansiones y mejoras en el monitoreo de otros factores ambientales, y demuestra el potencial de la tecnología de código abierto en la creación de soluciones efectivas y accesibles.

## Conclusiones individuales:

**Iñigo Orozco Rodriguez:**

En resumen, este proyecto ha logrado una integración efectiva de componentes electrónicos y tecnologías web para crear tanto una página web, como un prototipo físico. La combinación de

Arduino, Flask, Dash y Github (python) ha permitido la captura precisa de datos y su visualización interactiva. El servidor Flask facilita la comunicación entre el sistema y la base de datos, asegurando la persistencia de registros para futuros análisis. Este proyecto sienta las bases para aplicaciones más amplias en el campo de conexión de las cosas al internet.

**Franciso Javier Davy:**

En conclusión, el proyecto nos ha otorgado herramientas útiles para el uso de tecnologías que se utilizan y probablemente no sabíamos como se creaban o se utilizaban. Se inició entendiendo los conceptos básicos del campo de conexión de (IOT). Posteriormente conocimos los diferentes sensores que íbamos a utilizar. Además se aventuró en diferentes servidores y softwares como MySQL que fue nuestra base de datos y un codespace como Github.
Una vez con estos conceptos visualizados nos adentramos a integrar un prototipo físico y una página web..
Finalmente este proyecto nos percató del uso de dispositivos en nuestra vida diaria y sus futuras aplicaciones.

**Jorge Siegrist Rivera**

Este proyecto ha sido todo un éxito. Logramos mezclar distintas tecnologías y componentes electrónicos para hacer una página web y un prototipo físico súper innovadores. Usamos muchas herramientas nuevas como Arduino, Flask, Dash y Github, y también MySQL para nuestra base de datos. Todo esto hizo que capturar y mostrar los datos fuera algo muy fluido y fácil de manejar. Además, aprendimos muchísimo sobre tecnologías nuevas y cómo se aplican en el mundo real, especialmente en el Internet de las Cosas (IoT). Este proyecto nos ayudo mucho para entender cómo funcionan los sensores y cómo los podemos aplicar en la vida real. En resumen, este proyecto nos dejó bien parados para crear cosas ya con el conocimiento para poder crear e innovar usando las tecnologías avanzadas en el futuro en el mundo de la tecnología conectada.

**Tomas Garcia Estebarena:**

Este proyecto destaca por su enfoque innovador en la creación de aplicaciones web y la conexión de las cosas al internet. La combinación de sensores, comunicación inalámbrica y una interfaz web interactiva gracias a las aplicaciones y conocimientos utilizados, proporcionan un sistema efectivo y versátil. La integración fluida mediante un servidor Flask y una base de datos MySQL, establece una sólida base para futuras aplicaciones.

**Alejandro Ostos Rojí**

En conclusión, este proyecto nos deja muchas enseñanzas, desde cómo funcionan las bases de datos, hasta cómo podemos crearlas e interactuar con ellas. Al igual aprendimos a conectar todos estos códigos por medio de servidores para poder crear páginas web. Una de las cosas más retadoras a lo largo de este reto fue la creación de los códigos en python y toda la lógica que estos conllevan por detrás para poder ser conectados al internet.

## Referencias:


- Elmasri, R., & Navathe, S. B. (2016). Fundamentals of database systems (7th ed.). Pearson Education.
- Codd, E. F. (1970). A relational model of data for large shared data banks. Communications of the ACM, 13(6), 377-387.
- Wikipedia. (n.d.). Base de datos. Wikipedia, La enciclopedia libre. https://es.wikipedia.org/wiki/Base_de_datos
- Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. Computer Networks, 54(15), 2787-2805.
- Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645-1660.
- Wikipedia. (n.d.). Internet de las cosas. Wikipedia, La enciclopedia libre. https://es.wikipedia.org/wiki/Internet_de_las_cosas
- Baresi, L. (2017). Arduino: Guía completa para principiantes. Packt Publishing.
- Arduino. (n.d.). Arduino. Arduino. https://www.arduino.cc/
- Wikipedia. (n.d.). Arduino. Wikipedia, La enciclopedia libre. https://es.wikipedia.org/wiki/Arduino
- Elliott, C. (2019). SQL in a nutshell (8th ed.). O'Reilly Media.
- Date, C. J. (1987). SQL and relational theory: How to write correct SQL code. ACM Computing Surveys, 19(1), 1-36.
- Página web: Wikipedia. (n.d.). SQL. Wikipedia, La enciclopedia libre. https://es.wikipedia.org/wiki/SQL