IOT: Presentacion Final

Alejandro Ostos Roji IID | A01552398

Thomas Garcia Estebarena IID | A01662503

Iñigo Orozco Rodriguez IID A01783125

Jorge Siegrist IID | A01782873

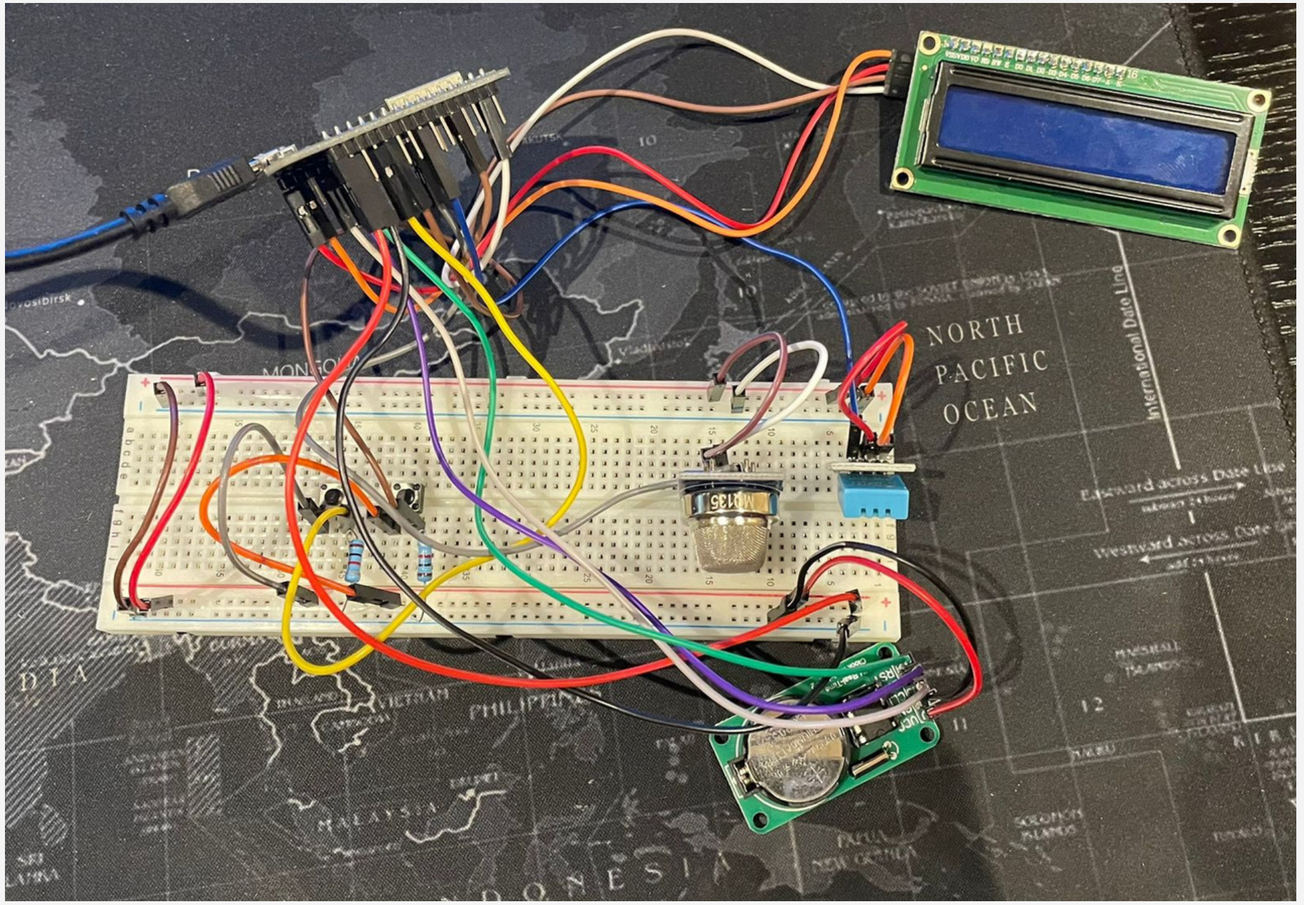
Francisco Javier Romero IID | A01783170

Agenda

- Definicion del reto
- Circuito fisico
- Codigo en Arduino
- Codigo de Github (Servidor FLASK)
- Codigo de Github (showdata)
- Arquitectura del producto

Definicion del reto

Circuito Fisico



Co
Ar

```
#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: ");
```

Codigo de 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
        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', 'IljS3LDZZm', '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 datos se recibieron correctamente
```

Codigo de Github (Showdata)

```python
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 Time")),
                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)
```

Arquitectura del Producto

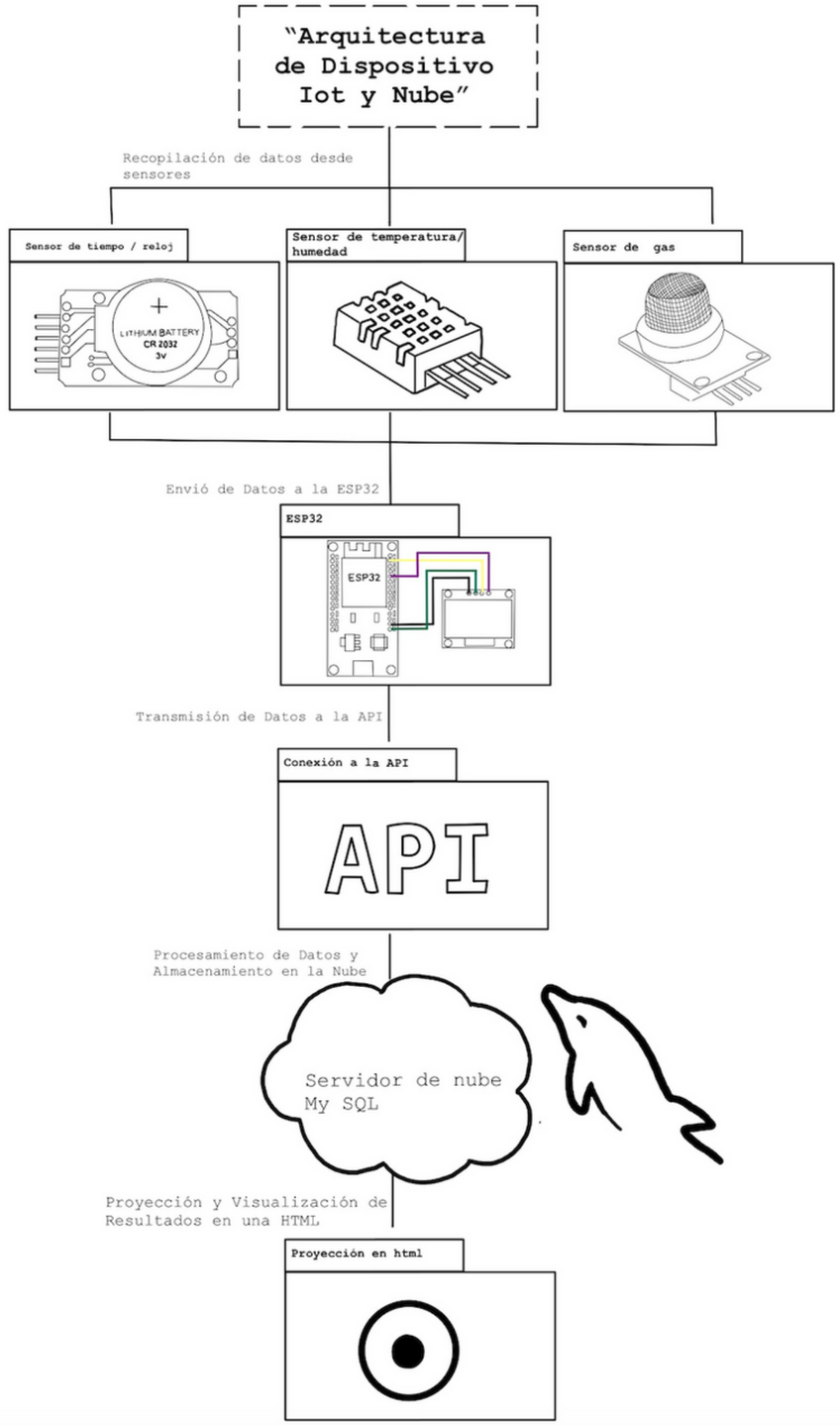
Paso 1: Recopilación de Datos desde Sensores

Paso 2: Envío de Datos a la ESP32

Paso 3: Transmisión de Datos a la API

Paso 4: Procesamiento de Datos y Almacenamiento en la Nube

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



Conclusión