



Tecnológico de Monterrey

Implementación del Internet de las Cosas

Reto - Aplicación completa IOT

Fecha de entrega: 18/10/2023

Alumnos

Alejandro Elías Slim	A01782184
Paulina Dodero Riba	A01782516
Giuliana Mosti Contreras	A01782192
Rodrigo Riveroll Dodero	A01782495

Índice:

Resumen del Proyecto.....	3
Introducción.....	3
Materiales.....	3
Desarrollo.....	5
Prototipo.....	5
Código de lectura de sensores.....	6
Base de datos en la nube.....	10
Conexión a un servicio RESTFUL.....	11
Aplicación WEB para visualización de datos.....	13
Arquitectura del Proyecto.....	16
Resultados.....	17
Conclusión General.....	17
Conclusiones Personales.....	18
Rodrigo Riveroll Dodero.....	18
Paulina Dodero Riba.....	18
Giuliana Mosti Contreras.....	19
Alejandro Elías Slim.....	19

Resumen del Proyecto

El proyecto de Implementación de Internet de las Cosas (IoT), se propone como una solución innovadora en el campo de la tecnología, desarrollando un sistema de monitoreo basado en el conjunto de tecnologías de hardware y software. Utilizando un microcontrolador ESP32 y diversos sensores, logramos recolectar y enviar datos de manera efectiva a una base de datos en la nube a través de una API RESTFUL desarrollada en Python. La visualización de estos datos se logró mediante una interfaz web desarrollada con Dash, permitiendo un seguimiento en tiempo real de las variables monitoreadas de forma gráfica. Este proyecto detecta, registra, almacena y muestra datos de sensores de humedad, temperatura, gas y tiempo.

Introducción

Durante estas 10 semanas estuvimos experimentando con el ecosistema de IoT. Al ser un tema nuevo requirió muchas prácticas y ejercicios para familiarizarnos con todo el entorno. Desde como hacer el cableado, como funciona el lenguaje para manejo de bases de datos - MySQL, la programación en Python, saber que es un API y cómo funciona, cómo hacer conexiones de bases de datos y cómo poder integrar estos sistemas a Alexa junto con diferentes Chatbots. Para todo esto utilizamos muchos softwares de modelación como Tinkercad y WOKWI. También utilizamos diversos sensores y al mismo tiempo fuimos manipulando bases de datos, aprovechando la nube y sus servidores, en donde nuestro proyecto acabaría viviendo para ser ejecutado más adelante. Todo esto con el fin de poder juntar todos los aprendizajes y posteriormente poder integrarlo en un sistema que simule un refrigerador inteligente. Este sistema buscaba hacer una conexión exitosa entre 4 sensores con el microcontrolador ESP32 para así poder generar datos, recolectarlos y mandarlos a una base de datos en la nube y por último poder graficarlos en una página web, a través del lenguaje HTML.

Materiales

En esta sección, se proporcionará una descripción de los materiales que fueron utilizados para la realización de nuestro proyecto. Cada uno de estos componentes desempeña un papel fundamental en la implementación y funcionalidad de la solución.

Material	Descripción / USO	Imagen
-----------------	--------------------------	---------------

Protopboard	Se utilizó para conectar todos los elementos de nuestros circuitos	
Cables (macho - hembra), (macho - macho) y (hembra - hembra)	Se utilizaron para conectar componentes electrónicos entre sí	
ESP - 32	Habilita la conectividad inalámbrica, siendo el puente entre los sensores y la base de datos en la nube	
Sensor DHT11	Mide la humedad y temperatura ambientales para mantener condiciones óptimas	
Reloj DS1302	Rastrea el tiempo en tiempo real, esencial para registrar datos con precisión temporal	
Gas MQ135	Detecta niveles de diferentes gases en el ambiente, útil para monitorear la calidad del aire	
Display LCD 16 IC2	Muestra la información recopilada en tiempo real, ofreciendo un vistazo rápido a las condiciones actuales	

2 Push Buttons	Permite al usuario interactuar con el sistema, como cambiar la información mostrada en el display	
----------------	---	---

Tabla 1: Materiales utilizados

Desarrollo

Prototipo

1. Nuestro primer paso fue conectar los sensores al ESP32.

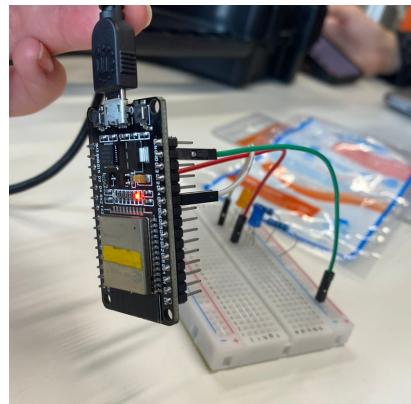


Imagen 1: Conexión de los sensores a la ESP32

2. Una vez conectados, se conectó el display.

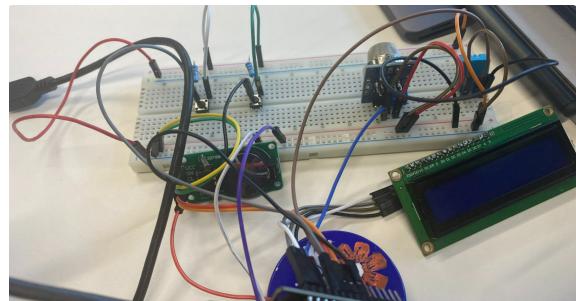


Imagen 2: Conexión del display

3. Para ver reflejados los datos, se utilizó el código de arduino junto con la protoboard, donde se imprimió un mensaje para ver que estuviera funcionando correctamente.



Imagen 3: Prueba de display

Código de lectura de sensores

Una vez tenido el circuito conectado y funcionando correctamente, desarrollamos un código que nos permitiera leer y “extraer” los datos de los sensores en Arduino IDE. Este código incluye las librerías:

Wire.h	Permite la comunicación entre dispositivos electrónicos, como sensores y pantallas LCD,
LiquidCrystal_I2C.h	Se utiliza para controlar pantallas LCD de caracteres, específicamente para pantallas LCD que se conectan a través de I2C y simplifica la comunicación con estas pantallas.
DFRobot_DHT11.h	Proporciona las funciones necesarias para leer los datos del sensor de humedad y temperatura DHT11
WiFi.h	Establece la conexión WiFi
HTTPClient.h	Es utilizada para la comunicación con servidores web
ArduinoJson.h	Facilita la manipulación y generación de datos en formato JSON en placas Arduino y ESP32.
MQ135.h	Permite leer los valores de concentración en PPM de gases del sensor MQ135
Ds1302.h	Permite establecer y leer la fecha y la hora del reloj en tiempo real con el sensor/módulo Ds1302.

Tabla 2: librerías del código de lectura de sensores

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DFRobot_DHT11.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <MQ135.h>
#include <Ds1302.h>
```

Una vez incluidas, se establecen y configuran objetos y variables que serán utilizados posteriormente en el programa para controlar la pantalla LCD, interactuar con sensores y realizar conexiones de red para el envío de datos. También se definen los días de la semana en un arreglo para su uso en la aplicación.

```
//MQ135 gasSensor(MQ135_PIN);
LiquidCrystal_I2C lcd(0x27, 16, 2);
DFRobot_DHT11 DHT;
Ds1302 rtc(PIN_RST, PIN_CLK, PIN_DAT);
String date_time;

const int buttonPin1 = 33;
const int buttonPin2 = 14;
int displayMode = 0; // 0: Temp, 1: Humidity, 2: Gas, 3: Time
const char* ssid = "Tec-Contingencia";
const char* password = "";
const char* apiEndpoint =
"https://ominous-giggle-69g47p969jjjc4wrx-5000.app.github.dev/sensor_data";

const static char* WeekDays[] = {
"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"
};
```

Posteriormente, se establece la conexión WiFi usando el ID y la contraseña definidos en la sección anterior, esto mediante la función setupWifi().

```
void setupWifi() {
Serial.begin(9600);
Serial.print("Connecting to WiFi");
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(1000);
Serial.print(".");
}
Serial.print(" Connected: ");
Serial.println(WiFi.localIP());
}
```

Después, se inicializan y configuran varios componentes, como la pantalla LCD, la comunicación serial, los pines de entrada para botones, el módulo DS1302 y se llama a la función setupWifi().

```
void setup() {
```

```

lcd.init(); // Pass your I2C address as an argument if it's not the default
lcd.backlight();
Serial.begin(115200);
pinMode(buttonPin1, INPUT);
pinMode(buttonPin2, INPUT);
rtc.init();

Ds1302::DateTime now;
rtc.getDateTime(&now);

String date_time = "20" + String(now.year) + "-" +
((now.month < 10) ? "0" : "") + String(now.month) + "-" +
((now.day < 10) ? "0" : "") + String(now.day) + " " +
((now.hour < 10) ? "0" : "") + String(now.hour) + ":" +
((now.minute < 10) ? "0" : "") + String(now.minute) + ":" +
((now.second < 10) ? "0" : "") + String(now.second);

setupWifi();
}

```

La función sendData funciona con los datos del sensor de temperatura, humedad, gas y tiempo y los envía al API (Application Programming Interface); esto funciona mediante la creación de un objeto HttpClient con el cual se inicia la solicitud HTTP POST al punto final de la API(sensor_data). Después se construye un objeto en formato JSON con los datos recibidos para facilitar su envío y lectura, para después realizar el método POST de http y enviar los datos de manera correcta.

```

void sendData(float temperature, float humidity, int mq135Value, String date_time)
{
  Serial.print("Sending data to API: ");

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

  StaticJsonDocument<300> doc;

  doc["temperature"] = temperature;
  doc["humidity"] = humidity;
  doc["mq135Value"] = mq135Value;
  doc["date_time"] = date_time;

  String json;
  serializeJson(doc, json);
  Serial.println("Sending JSON: " + 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);

// Aquí puedes agregar lógica adicional según la respuesta del servidor
// Por ejemplo, verificar si la solicitud fue exitosa, etc.
} else {
Serial.print("Error code: ");
Serial.println(httpResponseCode);

// Aquí puedes agregar manejo de errores, como intentar nuevamente o notificar
problemas.
}

http.end();
}

```

Por último se declara la función loop(). Esta es la sección más importante del código ya que es la encargada de leer los datos de los sensores, mostrarlos en el display según elija el usuario por medio de los push buttons, y llamar a la función sendData() con los valores obtenidos de los sensores. Esta sección, al ser un loop, se repite constantemente con un delay de 2000 ms hasta frenar el código.

```

void loop() {
// Tu código para el manejo de los botones y sensores aquí
lcd.clear();
Ds1302::DateTime now;
rtc.getDateTime(&now);

String date_time = "20" + String(now.year) + "-" +
((now.month < 10) ? "0" : "") + String(now.month) + "-" +
((now.day < 10) ? "0" : "") + String(now.day) + " " +
((now.hour < 10) ? "0" : "") + String(now.hour) + ":" +
((now.minute < 10) ? "0" : "") + String(now.minute) + ":" +
((now.second < 10) ? "0" : "") + String(now.second);

DHT.read(DHT11_PIN);
float temperature = DHT.temperature;
float humidity = DHT.humidity;

int mq135Value = analogRead(MQ135_PIN) / 100;

```

```

// El resto de tu función sendData permanece igual

if (digitalRead(buttonPin1) == LOW && digitalRead(buttonPin2) == LOW) {
    DHT.read(DHT11_PIN);
    lcd.setCursor(0, 0);
    lcd.print("Temp:");
    lcd.print(temperature);
}

else if (digitalRead(buttonPin1) == HIGH && digitalRead(buttonPin2) == LOW) {
    DHT.read(DHT11_PIN);
    lcd.setCursor(0, 0);
    lcd.print("Humi: ");
    lcd.print(humidity);
    lcd.print("%");
}

else if (digitalRead(buttonPin1) == LOW && digitalRead(buttonPin2) == HIGH) {
    lcd.setCursor(0, 0);
    lcd.print("CO2 (PPM): ");
    lcd.print(mq135Value);
}

else if (digitalRead(buttonPin1) == HIGH && digitalRead(buttonPin2) == HIGH) {
    lcd.print(date_time);
}

sendData(temperature, humidity, mq135Value, date_time);
delay(2000); // Ajusta este delay según tus necesidades
}

```

Base de datos en la nube

Con el fin de tener un “lugar” en la nube en donde se pudieran almacenar los datos de las lecturas de los 4 sensores, creamos una base de datos en la nube usando Free MySQL Hosting net. Esta usa un servidor web, al igual que un nombre, usuario y contraseña de acceso creadas por el sistema.

Your new database is now ready to use.

To connect to your database use these details

Server: sql10.freemysqlhosting.net
Name: sq10652554
Username: sq10652554
Password: Al989QAABC
Port number: 3306

Imagen 4: Datos de MySQL

Una vez creada, definimos los nombres de las columnas, al igual que el tipo de valor que

estas iban a almacenar. Establecimos la columna ID (número identificador) como Primary Key ya que es el identificador de los valores, éste también incrementa +1 de manera automática.

	Editar	Copiar	Borrar	id	humidity	temperature	mq135Value	date_time
<input type="checkbox"/>				300	52	20	40	2023-10-10 01:28:40
<input type="checkbox"/>				301	52	20	40	2023-10-10 01:28:45
<input type="checkbox"/>				302	52	20	40	2023-10-10 01:28:50
<input type="checkbox"/>				303	52	20	40	2023-10-10 01:28:56
<input type="checkbox"/>				304	52	20	40	2023-10-10 01:29:01
<input type="checkbox"/>				305	52	20	40	2023-10-10 01:29:06
<input type="checkbox"/>				306	52	20	40	2023-10-10 01:29:11
<input type="checkbox"/>				307	52	20	40	2023-10-10 01:29:17
<input type="checkbox"/>				308	52	20	40	2023-10-10 01:29:22
<input type="checkbox"/>				309	51	20	40	2023-10-10 01:29:28
<input type="checkbox"/>				310	51	20	40	2023-10-10 01:29:33
<input type="checkbox"/>				311	51	20	40	2023-10-10 01:29:38

Imagen 5: Sección de la tabla dht_data en MySQL

Conexión a un servicio RESTFUL

Antes de que los datos estén en la nube, es necesario crear una aplicación web Flask que ofrezca un servicio RESTful para obtener y almacenar dichos datos. Ésta funciona con el siguiente código:

```
from flask import Flask, request, render_template, jsonify
from datetime import date, datetime, timedelta
import mysql.connector
from flask import Flask, render_template
import matplotlib.pyplot as plt
from io import BytesIO
import base64

app = Flask(__name__)
```

Se importan las siguientes librerías módulos y se crea una instancia de la aplicación Flask, la cual se utilizará para definir rutas, vistas y manejar solicitudes y respuestas HTTP.

flask	Se importan varios módulos de Flask y otros, necesarios para crear una aplicación web. Esto incluye el módulo Flask, que se utiliza para crear la aplicación, así como otros módulos para manejar solicitudes HTTP, renderizar plantillas HTML y responder con datos JSON
datetime	Estos módulos se utilizan para trabajar con fechas y horarios en la aplicación web

mysql.connector	Se utiliza para conectarse a una base de datos MySQL y permite almacenar, recuperar y manipular datos.
matplotlib.pyplot	Se usa para visualizar datos y graficaciones
io	Se importa el módulo BytesIO para la manipulación de datos binarios.
base64	Se utiliza para codificar y decodificar datos en formato base64.

Tabla 3: Librerías utilizadas en el código de conexión a un servicio RESTFUL

La función siguiente encapsula la lógica de conexión con la base de datos de MySQL. Además, crea dos objetos “cnx” y “cursor”, los cuales representan la conexión a la base y se utilizan para ejecutar consultas y comandos respectivamente.

```
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)
```

Después, en el método GET se crea la conexión a la base de datos con los parámetros proporcionados (usuario, nombre, contraseña, etc...). Después ejecuta el query que selecciona los datos de la base para poder usarlos posteriormente; mientras el método POST, usando la ruta “sensor_data”, se encarga de recibir los datos enviados desde el código de Arduino IDE , crear una conexión con la base de datos en la nube y ejecutar el query que inserta los datos en la columna de la base de datos que le corresponde a cada valor. Finalmente envía un mensaje de respuesta dependiendo de el éxito de la operación (datos recibidos/datos no recibidos)

```
@app.route('/', methods=['GET'])
def get_sensor_data():
    # Create a connection to the database
    cnx, cursor = createConnection(
        'sql10652554', 'sql10652554', 'AI989QAABC', 'sql10.freemysqlhosting.net', '3306')
    # Query the database
    query = ("SELECT * FROM dht_data")
    # Execute the query
    cursor.execute(query)
    # Get the data
    data = cursor.fetchall()
```

```

# Close the connection
cursor.close()
cnx.close()
print(data)
# Obtener los valores de x e y desde los datos
x = [item[0] for item in data]
y1 = [item[1] for item in data]
y2 = [item[2] for item in data]
y3 = [item[3] for item in data]
y4 = [item[4] for item in data]

@app.route('/sensor_data', methods=['POST'])
def receive_sensor_data():
if request.headers['Content-Type'] == 'application/json':
data = request.json

humidity = data.get('humidity')
temperature = data.get('temperature')
date_time = data.get('date_time')
mq135Value = data.get('mq135Value')

print("Received humidity:", humidity)
print("Received temperature:", temperature)
print("Received mq135Value:", mq135Value)
print("recieved date_time:", date_time)

cnx, cursor = createConnection(
'sql10652554', 'sql10652554', 'AI989QAABC', 'sql10.freemysqlhosting.net', '3306')

add_data = (
"INSERT INTO dht_data (humidity, temperature, mq135Value, date_time) VALUES (%s,
%s, %s, %s)")
cursor.execute(add_data, (humidity, temperature, mq135Value, date_time))
cnx.commit()
cursor.close()
cnx.close()
print

return 'Data received successfully.', 200
else:
return 'Invalid content type. Expected application/json.', 400

```

Aplicación WEB para visualización de datos

Para poder visualizar de forma gráfica los datos almacenados en la base en la nube, creamos una aplicación web usando el siguiente código:

```

import mysql.connector
import pandas as pd
import dash
from dash import dcc, html
import plotly.express as px
from dash.dependencies import Input, Output
import time

```

Primero, como en los códigos anteriores, se importan las siguientes bibliotecas y módulos:

mysql.connector	Se utiliza para conectarse a una base de datos MySQL y permite almacenar, recuperar y manipular datos.
pandas	Es utilizada para el análisis y manipulación de datos, y es comúnmente utilizada para trabajar con conjuntos de datos tabulares.
dash	Es el framework que se utiliza para crear la aplicación web interactiva.
plotly.express	Biblioteca de visualización de datos que proporciona una interfaz sencilla para crear gráficos interactivos.
time	Se utiliza para trabajar con el tiempo y la sincronización

Tabla 4: Librerías utilizadas en el código de la aplicación web

Después, creamos una función que selecciona los últimos datos ingresados y los regresa.

```

def select_latest_data():
    try:
        cnx = mysql.connector.connect(user='sql10652554', database='sql10652554',
                                      password='AI989QAABC', host='sql10.freemysqlhosting.net', port='3306')
        cursor = cnx.cursor()

        # Consultar la base de datos para obtener los últimos datos
        query = ("SELECT * FROM dht_data ORDER BY id DESC LIMIT 100") # Puedes ajustar la
        cantidad de datos que deseas obtener

        cursor.execute(query)
        data = cursor.fetchall()
        cnx.close()
        cursor.close()

        return data

    except mysql.connector.Error as err:

```

```

# Manejo de errores
if err.errno == mysql.connector.errorcode.ER_ACCESS_DENIED_ERROR:
    print("Something is wrong with your user name or password")
elif err.errno == mysql.connector.errorcode.ER_BAD_DB_ERROR:
    print("Database does not exist")
else:
    print(err)
return None

```

A continuación creamos una función que recibe los datos, los convierte en un DataFrame (tipo de arreglo de los datos) para convertirlos en una gráfica de líneas usando Plotly Express y la regresa.

```

def plot_data():
    try:
        data = select_latest_data()

        if data:
            df = pd.DataFrame(data, columns=['id', 'humidity', 'temperature', 'mq135Value',
                                             'date_time'])
            fig = px.line(df, x='date_time', y=['humidity', 'temperature', 'mq135Value'],
                           labels={'humidity': 'Humedad', 'temperature': 'Temperatura', 'Gas': 'mq135Value'})
            fig.update_layout(
                title='Gráfica de Humedad, Temperatura y Gas',
                xaxis_title='Date Time',
                yaxis_title='Valores'
            )

            return fig
    except Exception as e:
        print(f"Error: {e}")
        return None

```

Por último, se crea una aplicación web Dash que muestra un gráfico interactivo que se actualiza automáticamente cada 3 segundos, utilizando datos generados por la función `plot_data()`. La aplicación se ejecuta localmente en un servidor de desarrollo.

```

app = dash.Dash(__name__)

app.layout = html.Div([
    html.Div(
        children=[
            html.H1("Temperature, Humidity and Gas", style={'text-align': 'center'}),

```

```

html.P("Esta aplicación muestra una gráfica de linea con datos de humedad y
temperatura. Los datos fueron generados con un código de python que generó 100
datos entre 0 y 100 tanto para temperatura como para humedad."),
dcc.Graph(id='live-update-graph', figure=plot_data()),
dcc.Interval(
id='interval-component',
interval=3 * 1000, # Actualizar cada 3 segundos (en milisegundos)
n_intervals=0
)
])
])
])

@app.callback(
Output('live-update-graph', 'figure'),
Input('interval-component', 'n_intervals')
)

def update_graph(n):
return plot_data()

if __name__ == '__main__':
app.run_server(debug=True)

```

Arquitectura del Proyecto

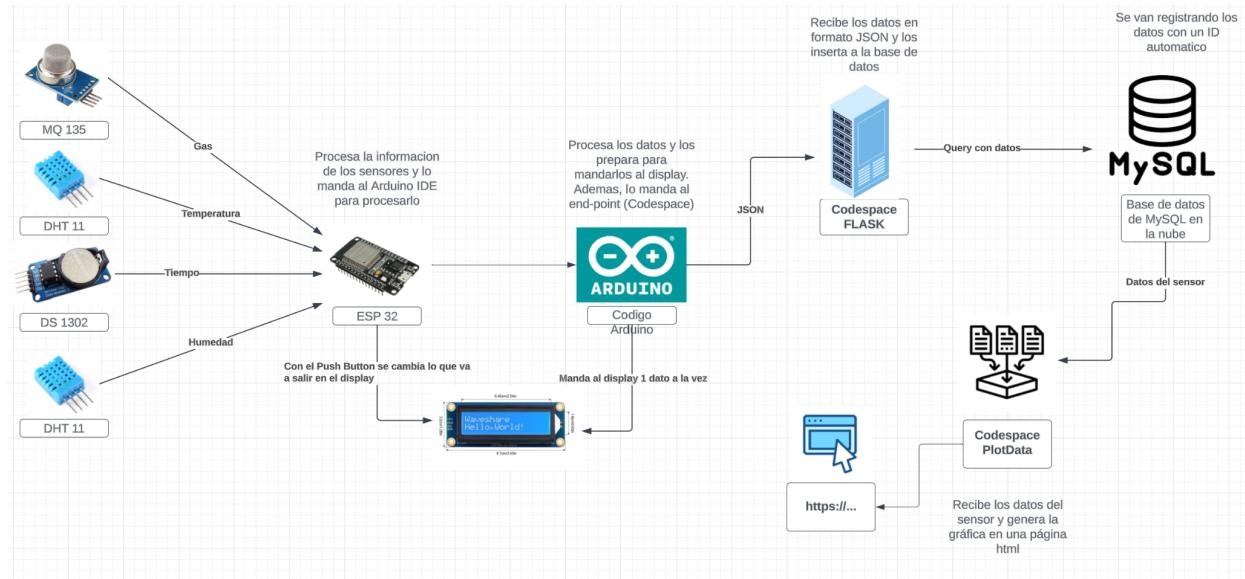


Imagen 10: Arquitectura del producto

El proyecto se inicia con la conexión de sensores MQ 135, DHT 11 y DS 1302 a un ESP 32, el cual procesa y envía sus datos tanto al display como a un programa en Arduino, visualizando simultáneamente en un display LCD16x2. Luego, estos datos se manejan en Arduino IDE, desde donde se preparan y se remiten al endpoint "sensor_data" en Codespace

FLASK, utilizando formato JSON. FLASK, actuando como framework, envía consultas con estos datos a la nube de MySQL, donde se registran con un ID automático. Finalmente, los datos se transfieren a PlotData, que emplea librerías como Dash y Plot para generar un HTML visualmente interactivo con gráficas detalladas de la información recopilada.

Resultados

Link al repositorio de Github: https://github.com/paudodero/SITUACION_IOT

En este repositorio se encuentran todos los códigos y documentos requeridos para el funcionamiento de nuestro proyecto.

En la siguiente imagen se puede observar la gráfica de los datos recibidos por los sensores en el momento de hacer nuestra presentación final, la cual se encuentra adjunta debajo de la imagen.

Temperature, Humidity and Gas

Esta aplicación muestra una gráfica de líneas con datos de humedad y temperatura. Los datos fueron generados con un código de python que generó 100 datos entre 0 y 100 tanto para temperatura como para humedad.

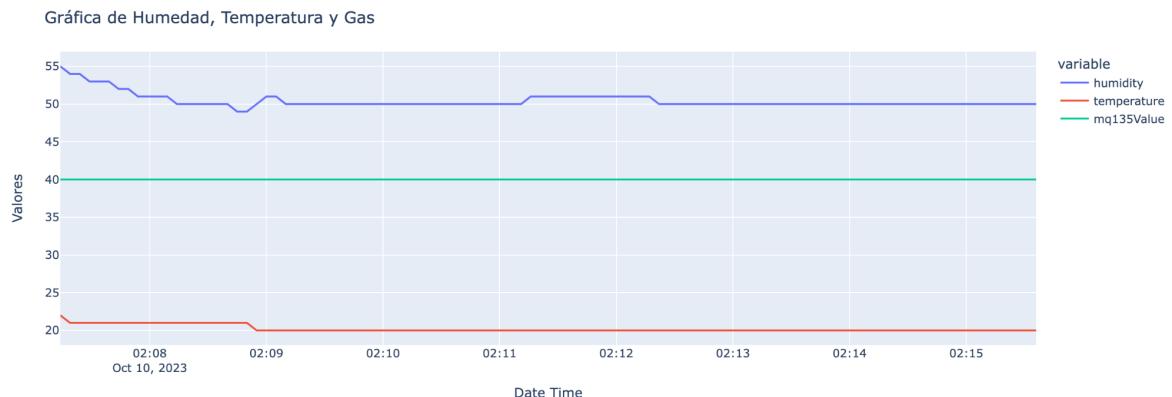


Imagen 11: del gráfico de la visualización de datos en el app Web.

Presentación final:

https://www.canva.com/design/DAFxQjIVg9c/4Hm_if9XxCHWR9WoqB7Ng/edit?utm_content=DAFxQjIVg9c&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Conclusión General

Como pudimos observar a lo largo del proyecto el IoT es un mundo, se requiere de grandes conocimientos, tanto de software como de hardware para darle un uso beneficioso, pero todo el trabajo tiene mucho sentido cuando nos damos cuenta de la cantidad de usos y aplicaciones

que tiene el Internet de las cosas. En lo personal: podemos usarlo para nuestros hogares (controlar luces, cerraduras y electrodomésticos), para nuestra salud (smartwatches que midan nuestro ritmo cardíaco, actividad física...) o inclusive para entretenimiento y lujo (experiencias de cine o escenario casero, controlado luces y sonido), en fin, el IoT en nuestra vida personal tiene y tendrá cada vez más usos. En un ambiente comercial, creemos que es aún más útil: Por ejemplo, en la agricultura, puede ofrecer datos vitales para maximizar los rendimientos y usar eficientemente los recursos; en negocios, facilita la gestión de almacenes e inventarios para optimización de costos; en el esquema de las ciudades inteligentes, potencia la monitorización eficiente del tráfico y puede medir la calidad del aire o en las mismas fábricas puede ser implementado para monitorear la actividad del personal y calidad de la producción.

Conclusiones Personales

Rodrigo Riveroll Dodero

Después de dos periodos completos he notado una gran diferencia en mi ya que esta clase me ha aportado muchas cosas nuevas que me han servido no solo para mi vida universitaria sino también para mi vida profesional. Creo que estuvo muy bien estructurado como poco a poco íbamos aprendiendo cosas que juntandolas iban construyendo nuestras habilidades para poder hacer la situación problema. Empezar con circuitos muy básicos y con uso básico de código en Arduino, C ++ y Python, hasta integrar chatbots como twilio a circuitos utilizando el ESP-32 y conectarlos con una base de datos en la nube fue algo que nunca pensé que iba a lograr. Además de todo esto me di cuenta que hay aplicaciones en la vida real de estos sistemas que pueden revolucionar muchas empresas e industrias. Esto yo lo he estado viviendo día a día en mi trabajo ya que aunque estoy en un entorno solo financiero se han ido implementando este tipo de tecnologías, junto con inteligencia artificial, y han logrado cambiar la forma de trabajar en la oficina. Además, hablando con el equipo que está llevando toda el área de implementación de chat GPT, pude entender todo lo que me decían ya que la clase me dio el conocimiento necesario para asimilar todos los conceptos. La verdad es que me quedo muy contento con mi desempeño ya que aprendí mucho y el resultado de la situación problema fue muy bueno.

Paulina Dodero Riba

Al terminar este reporte de entrega final me puedo dar cuenta de lo mucho que aprendí en estas 10 semanas de bloque de IoT, comenzando por la definición real del internet de las cosas; tema que creo que será de suma importancia en lo que me resta de carrera. Creo que pude fortalecer mis habilidades de programación en Python, al igual que en C ++ para Arduino y aprender SQL y manejo de bases de datos. Sin embargo, creo que lo que más me llevo del curso son las conexiones. Comenzando con el circuito, la conexión de los 4 sensores a la ESP32 y al display LCD fueron algo que nos costó como equipo. Al entender el acomodo de los pines es decir, cual es analógico, tierra o voltaje, logramos conectar el circuito de manera correcta y así hacerlo funcionar.

Por otro lado, la conexión de twilio al código de arduino y al celular me pareció, aunque no pudimos incluirlo en el reto, un tema muy interesante. Ver cómo funciona y en dónde podríamos aplicarlo fue un tema que me gustó aprender.

Por último, considero que la conexión de MySQL con un código, ya sea de arduino o flask o dash etc..., fue el tema que más me interesó, del que menos tenía conocimiento y del que más aprendí. Es un tema que considero de suma importancia en mi vida profesional, ya que hoy en día todo se maneja con datos en la nube y saber poder extraerlos, consultarlos, editarlos, analizarlos e interpretarlos o visualizarlos, como es que lo hicimos en nuestro proyecto, es una gran herramienta en la vida.

Giuliana Mosti Contreras

Estas 10 semanas fueron muy valiosas, y me hacen pensar que tengo que empezar a cuestionarme más las cosas, ya que el Iot siempre había estado alrededor de mí pero nunca me había cuestionado como funcionaba. Todo lo que aprendí fue nuevo y muy valioso, me impresionó como un aparato como el ESP 32 puede conectarse a la red WI-FI y hacer que otros dispositivos también se conecten entre sí. Claramente, también conocía el término de “En la nube” pero jamás lo había entendido al 100%, ya que no había tenido la oportunidad de hacerlo por mí misma y ver cómo funcionan las rutas de los datos para hacer las páginas web. También esto hizo que me diera cuenta de los posibles inconvenientes que pueden suceder y cómo repararlos, además que me ayudó mucho a mejorar mis habilidades de programación. Fue increíble poder conectar tantas cosas entre sí, ya que teníamos 5 componentes en el circuito funcionando a la vez de 3 códigos distintos en la nube y poder

aprender y estar involucrada en cada paso es muy importante para las competencias que necesito tener en mi desarrollo académico.

Por otro lado, me sorprendió mucho ver cómo se pueden hacer las notificaciones desde Twilio y mandarlas a cualquier celular. Lo que más me llevo es la programación en Mysql, se me hizo demasiado útil para mi vida y divertido de hacer, ya que siempre he sido muy consciente que lo mejor que puedes hacer en tu vida profesional es analizar los datos.

Por último, me encantó poder hacer un proyecto que sea escalable a cualquiera de los usos que mencionamos en el salón y un millón de casos más.

Alejandro Elías Slim

Esta clase ha sido una experiencia importante para mí carrera universitaria, ofreciéndome una visión para diversos temas que también serán importantes para mi carrera profesional. Si tuviera que hacer un desglose cronológico y conciso de los aprendizajes que he adquirido durante el desarrollo del proyecto, sería el siguiente. Comenzando con el hardware, me familiaricé con la arquitectura y las capacidades del ESP32, aprendiendo a integrar los 3 sensores de manera efectiva. Después, la programación en Arduino me proporcionó insights prácticos en cómo las conexiones de hardware se traducen en funcionalidades digitales. Esto también me permitió comprender más profundamente el papel que desempeñan el display (mostrando los resultados) y los botones (permitiéndonos cambiar los datos mostrados). En cuanto a la integración de sistemas, la utilización de una API para enlazar Arduino con Flask me enseñó no sólo la usabilidad que tienen las APIs, sino también su importancia estratégica en la arquitectura de software. Posteriormente, la implementación de una base de datos MySQL en la nube me permitió tener una mejor comprensión sobre el manejo de servidores y las ventajas que ofrecen en términos de accesibilidad (trabajo en equipo) y escalabilidad. Finalmente, el uso de Dash fue crucial para desarrollar una interfaz web que visualiza de manera efectiva los datos recopilados por los sensores. En resumen, este proyecto ha sido una experiencia muy padre, que me ha equipado con habilidades tanto en hardware como en software, y más importante aún, me ha enseñado cómo aprovechar estos elementos para construir una solución de IoT exitosa.

