Escalado de Plataforma IoT desde la Capa Física a la Capa de Transporte deDatos con Almacenamiento en MySQL y creación de Dashboard utilizando Node-Red para mostrar los datos en gráficos.

Nombre del Estudiante: Francisco TORO GOITEA

DNI: 41.117.371

Asignatura: Tecnicatura Superior en Desarrollo del Software

Comisión: 2

Profesor: Alejandro Luis MAINERO y Agustina APERLO

Evidencia: 3

Fecha de entrega: 03 de noviembre, 2024

Índice

1. Introducción

Este documento describe el desarrollo e implementación de una plataforma IoT escalable desde la capa física hasta una capa de visualización de datos en tiempo real. En la primera fase del proyecto, se estableció la transmisión de datos desde microcontroladores ESP32 hacia una base de datos MySQL mediante el protocolo MQTT. En esta nueva fase, se ha escalado la solución para incluir un dashboard en Node-RED, permitiendo la visualización dinámica de los datos de sensores en indicadores y gráficos en tiempo real.

2. Objetivos del Proyecto

El objetivo principal es implementar un sistema IoT escalable y visualmente accesible. En esta etapa avanzada del proyecto, se busca no solo transmitir los datos desde los microcontroladores ESP32 a un servidor, sino también crear una interfaz de usuario en Node-RED donde los datos se presenten de forma clara y personalizable, con el objetivo de facilitar el análisis en tiempo real y la toma de decisiones.

3. Metodología de Trabajo

Se utilizó una metodología basada en una arquitectura modular y escalable, dividiendo el proyecto en las siguientes fases:

Configuración del microcontrolador ESP32: para la recolección de datos de sensores.
Transmisión de datos mediante MQTT: Implementación del protocolo MQTT para la transmisión eficiente de datos hacia el servidor.
Almacenamiento en MySQL: Desarrollo de un backend en Python que conecta los datos MQTT directamente a la base de datos.
Implementación de un Dashboard en Node-RED: Creación de un sistema de visualización en tiempo real donde el usuario puede ver y analizar los datos a través de gráficos y otros indicadores personalizados.

4. Implementación de la Solución

4.1 Configuración del ESP32

El microcontrolador ESP32 fue configurado para leer datos de sensores conectados a sus pines. La programación del ESP32 se realizó utilizando el entorno de desarrollo Wokwi, facilitando la integración con los sensores y la comunicación con la capa de transporte.

4.2 Implementación de la Capa de Transporte de Datos

Para la transmisión de datos desde los dispositivos ESP32 al servidor, se optó por el uso del protocolo MQTT debido a su baja latencia y eficiente uso de ancho de banda. Se instaló un servidor Mosquitto MQTT en el servidor para gestionar los mensajes entrantes de los dispositivos ESP32.

4.3 Desarrollo del Backend y Conexión con MySQL

El backend fue desarrollado en Python utilizando la biblioteca Paho-MQTT para suscribirse a los tópicos de datos enviados por los ESP32. Una vez recibidos, los datos son procesados y almacenados en una base de datos MySQL mediante la conexión directa. Se evitó el uso de PHPMyAdmin para maximizar la eficiencia del sistema.

4.4 Optimización del Sistema para Escalabilidad

Se diseñó el sistema con una arquitectura modular, lo que permite la fácil integración de nuevos nodos ESP32 y sensores sin comprometer la estabilidad. Esto se logró a través deuna configuración flexible del servidor Mosquitto MQTT y la optimización de la base de datos MySQL.

4.5 Creación de un Dashboard en Node-RED

En esta fase de escalamiento, se desarrolló un dashboard en Node-RED para visualizar los datos en tiempo real, con las siguientes funcionalidades:

- **Visualización de Datos en Gráficos e Indicadores**: Los datos se presentan en gráficos ui_chart, ui_gauge e ui_text, los cuales se actualizan en tiempo real para mostrar las tendencias de las métricas seleccionadas.
- **Notificaciones por umbral:** Aparece una notificación cuando el valor de temperatura o humedad, lleguen al valor de umbral asignado.
- **Panel para seleccionar métricas especificas en intervalos:** No lo he logrado, pero con más tiempo lo voy a agregar.

Anexos (Codigos – DML – JSON para importar Node-RED):

ESP32:

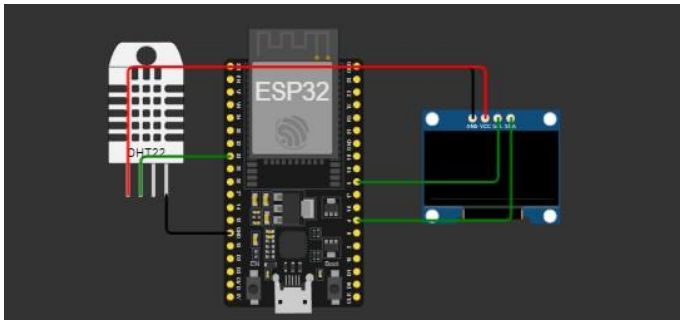
Link de Proyecto Wokwi: https://wokwi.com/projects/408634800407273473

Código sin link:

```
# Importaciones
import network
import time
import ssd1306
import dht
from machine import Pin, I2C
from umqtt.simple import MQTTClient

#Parametros de WiFi
SSID = "Wokwi-GUEST"
PASSWORD = ""

#Parametros del Broker MQTT
MQTT_BROKER = "broker.hivemq.com"
MQTT_PORT = 1883
MQTT_TOPIC = b"iot/sensors/dht22"

sensor = dht.DHT22(Pin(33))

#Conexión a Wifi
def connect_wifi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(SSID, PASSWORD)

    print('Conectando a la red...')
    while not wlan.isconnected():
        time.sleep(1)

    print('Conectado a la red:', wlan.ifconfig())

#Conexion al broker MQTT (HiveMQ)
def connect_mqtt():
    client = MQTTClient("esp32_client", MQTT_BROKER, port=MQTT_PORT) #Instancia
    client.connect()
    print(f"Conectado al broker MQTT {MQTT_BROKER}")
    return client
```

```
#Envio de datos del sensor al broker MQTT | Se pasa como parametro el client retornado de la función connect_mqtt
def publish_data(client):
    while True:
        try:
            sensor.measure()
            temp = sensor.temperature()
            hum = sensor.humidity()

            # Configuración del Display OLED SSD1306
            i2c_oled = I2C(scl=Pin(5), sda=Pin(4))
            oled_width = 128
            oled_height = 64
            oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c_oled)

            # Mostrar datos en el OLED
            oled.fill(0)
            oled.text('Temperature: {} C'.format(temp), 0, 0)
            oled.text('Humidity: {}%'.format(hum), 0, 10)
            oled.show()

            # Crear mensaje de datos
            message = f"Temperatura: {temp:.2f} °C, Humedad: {hum:.2f} %"
            print(f"Enviando mensaje: {message}")

            # Publicar el mensaje al tópico (topic) MQTT
            client.publish(MQTT_TOPIC, message.encode()) # encode para enviar en formato binario
            time.sleep(10)

        except Exception as e:
            print(f"Error al medir datos: {e}")
            time.sleep(5)  # Espera antes de volver a intentar

#Iniciamos ESP32
connect_wifi()
client = connect_mqtt() #Almacenamos en client, la instancia retornada de la función para luego usarlo en publish_data
publish_data(client)
```

Imagen del ESP32:



Script Python Back-End:

```
import paho.mqtt.client as mqtt
import mysql.connector
import json

#Datos de conexión a la base de datos MySQL que luego se utilizan en la función save_to_database
donde se hace la conexión. Formato JSON
db_config = {
    'user': 'root',
    'password': '420926',
    'host': 'localhost', #IP donde esta alojado el servidor
    'database': 'iot_data',
}

#Función que se ejecuta cuando se conecta al broker MQTT
def on_connect(client, userdata, flags, rc):
    print(f"Conectado al broker MQTT con codigo: {rc}")
    #Suscribirnos al topico (topic) donde subi los datos en el ESP32
    client.subscribe("iot/sensors/dht22")

#Función que se ejecuta cuando se recibe un mensaje desde el broker MQTT
def on_message(client, userdata, msg):
    print(f"Mensaje recibido: {msg.payload.decode()}") #Uso de metodo decode, ya que los datos
vienen en binario usando "enconde()" o b""

    #Convertir el mensaje recibido en datos de temp y humedad.
    message = msg.payload.de de()
    try:
    co
        data = message.split(",") #Separar o dividir en comas cada string
        temp = float(data[0].split(":")[1].strip().replace("°C", ""))
        hum = float(data[1].split(":")[1].strip().replace("%", ""))

        #Llamamos funcion para guardar los datos a la base de datos.
        save_to_database(temp,hum)
```

```
        print(f"Error procesando el mensaje: {e}")

#Función para guardar los datos en la base de datos, que luego se llama en la función on_message.
def save_to_database(temp,hum):
    try:
        #Conexión a base de datos
        connection = mysql.connector.connect(**db_config)
        cursor = connection.cursor()

        #Insertar los datos
        query = "INSERT INTO sensor_data (temperature, humidity) VALUES (%s, %s)"
        cursor.execute(query, (temp,hum))

        #Confirmar datos
        connection.commit()
        print(f"Datos guardados: Temperatura={temp}, Humedad={hum}")

    #Excepción dentro del bloque try, para capturar error.
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    #Cierre de conexión
    finally:
        cursor.close()
        connection.close()

#Configuración del cliente MQTT
client = mqtt.Client()

# Vincular las funciones de conexión y recepción de mensajes
client.on_connect = on_connect
client.on_message = on_message

#Conectar al broker MQTT.
client.connect("broker.hivemq.com", 1883, 60)

#Mantener la conexión y esperar los mensajes
client.loop_forever()
```

Base de datos MySQL:

Código para importar base de datos en MySQL Workbench:

```
-- MySQL dump 10.13  Distrib 8.0.38, for Win64 (x86_64)

--

-- Host: localhost           Database: iot_data

-- ------------------------------------------------------------------------------------------

-- Server version                8.0.39
```

```
/*!40101    SET  @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT  */;
/*!40101
            SET  @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101
            SET  @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
 /*!50503   SET  NAMES utf8 */;
 /*!40103   SET  @OLD_TIME_ZONE=@@TIME_ZONE */;
 /*!40103   SET  TIME_ZONE='+00:00' */;
 /*!40014   SET  @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
 /*!40014   SET  @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
 /*!40101   SET  @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
 /*!40111   SET  @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `sensor_data`
--

DROP TABLE IF EXISTS `sensor_data`;
/*!40101 SET @saved_cs_client                = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;CREATE TABLE
`sensor_data` (
   `id` int NOT NULL AUTO_INCREMENT,
   `timestamp` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
   `temperature` float DEFAULT NULL,
   `humidity` float DEFAULT NULL,PRIMARY
   KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `sensor_data`
--
```

```
LOCK TABLES `sensor_data` WRITE;

/*!40000 ALTER TABLE `sensor_data` DISABLE KEYS */;

INSERT INTO `sensor_data` VALUES (1,'2024-10-03 21:06:33',21.9,68),(2,'2024-10-03 21:06:44',-
12,34.5),(3,'2024-10-03 22:00:07',-12,34.5),(4,'2024-10-03 22:00:18',7.1,54),(5,'2024-10-03
22:15:15',2.8,35.5),(6,'2024-10-03 22:15:26',2.8,34.5),(7,'2024-10-03 22:15:51',2.8,34.5);

/*!40000 ALTER TABLE `sensor_data` ENABLE KEYS */;UNLOCK TABLES;

/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;


/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;

/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;

/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;

/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;

/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;

/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;


-- Dump completed on 2024-10-05 16:02:52
```

JSON para importar el flujo de Node-RED (Copiar e importar en Node-RED):


```
[{"id":"122012a1fbf7f480","type":"tab","label":"Flujo
1","disabled":false,"info":"","env":[]},{"id":"11730537ccd897d6","type":"inject","z":"122012a1fbf7f480","name":"intervalo","props":[{"p":"payload"},{"p":"topic","
vt":"str"}],"repeat":"5","crontab":"","once":false,"onceDelay":"1","topic":"","payload":"","payloadType":"date","x":400,"y":300,"wires":[["282bda7746ecf9a4"]]},{"
id":"282bda7746ecf9a4","type":"function","z":"122012a1fbf7f480","name":"Query Select","func":"msg.topic = \"SELECT temperature, humidity, timestamp FROM
sensor_data ORDER BY timestamp DESC LIMIT 1\";\nreturn
msg;\n\n","outputs":1,"timeout":0,"noerr":0,"initialize":"","finalize":"","libs":[],"x":590,"y":300,"wires":[["400994a684c7c749"]]},{"id":"400994a684c7c749","type
":"mysql","z":"122012a1fbf7f480","mydb":"a3a9bcc1eef4525f","name":"mysql","x":810,"y":300,"wires":[["92636547c2858e4c","49173326088104c8","dedf266a07697ffc","1d45
27d0d3bd5f29","4d30507e2313512b"]]},{"id":"f9828c47790f6fc1","type":"ui_text","z":"122012a1fbf7f480","group":"b7c01e510df0d097","order":1,"width":"0","height":"0"
,"name":"Temperatura","label":"Temperatura","format":"{{msg.payload.temperature}}","layout":"row-
spread","className":"","style":false,"font":"","fontSize":16,"color":"#000000","x":1390,"y":140,"wires":[]},{"id":"ef9f9085f7eda3b6","type":"ui_text","z":"122012a
1fbf7f480","group":"b7c01e510df0d097","order":3,"width":"0","height":"0","name":"Humedad","label":"Humedad","format":"{{msg.payload.humidity}}","layout":"row-
spread","className":"","style":false,"font":"","fontSize":16,"color":"#000000","x":1400,"y":500,"wires":[]},{"id":"e4b46db033c72b31","type":"ui_gauge","z":"122012
a1fbf7f480","name":"Gauge
Humedad","group":"b7c01e510df0d097","order":4,"width":"0","height":"0","gtype":"gage","title":"Humedad","label":"units","format":"{{msg.payload.humidity}}","min":
"1","max":"100","colors":["#00b500","#e6e600","#ca3838"],"seg1":"","seg2":"","diff":false,"className":"","x":1390,"y":580,"wires":[]},{"id":"915de1e7e86b9a97","ty
pe":"ui_gauge","z":"122012a1fbf7f480","name":"Gauge
Temperatura","group":"b7c01e510df0d097","order":2,"width":"0","height":"0","gtype":"gage","title":"Temperatura","label":"units","format":"{{msg.payload.temperatur
e}}","min":"1","max":"100","colors":["#00b500","#e6e600","#ca3838"],"seg1":"","seg2":"","diff":false,"className":"","x":1340,"y":40,"wires":[]},{"id":"92636547c28
58e4c","type":"function","z":"122012a1fbf7f480","name":"format data gauge-text","func":"const temperature = msg.payload[0].temperature;\nconst humidity =
msg.payload[0].humidity;\nmsg.payload = { temperature, humidity };\nreturn
msg;\n\n","outputs":1,"timeout":0,"noerr":0,"initialize":"","finalize":"","libs":[],"x":1060,"y":300,"wires":[["f9828c47790f6fc1","915de1e7e86b9a97","ef9f9085f7ed
a3b6","e4b46db033c72b31"]]},{"id":"72de9282c9029890","type":"ui_chart","z":"122012a1fbf7f480","name":"","group":"0b883c9bb34f389c","order":1,"width":"0","height":
"0","label":"Chart Temperatura","chartType":"line","legend":"true","xformat":"HH:mm:ss","interpolate":"linear","nodata":"Temperatura","dot":false,"ymin":"-
20","ymax":"80","removeOlder":1,"removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor":false,"useUTC":false,"colors":["#1f77b4","#aec7e8","#ff7f
0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferentColor":false,"className":"","x":1410,"y":260,"wires":[[]]},{"id":"491733
26088104c8","type":"function","z":"122012a1fbf7f480","name":"function chart temperature","func":"// Recupera el array de objetos desde msg.payload\nconst
temperature = msg.payload[0].temperature;\nmsg.payload = temperature;\nreturn
msg;\n\n\n","outputs":1,"timeout":0,"noerr":0,"initialize":"","finalize":"","libs":[],"x":1050,"y":240,"wires":[["72de9282c9029890"]]},{"id":"dedf266a07697ffc","t
ype":"function","z":"122012a1fbf7f480","name":"fuction chart humidity","func":"// Recupera el array de objetos desde msg.payload\nconst humidity =
msg.payload[0].humidity;\nmsg.payload = humidity;\nreturn
msg;\n\n","outputs":1,"timeout":0,"noerr":0,"initialize":"","finalize":"","libs":[],"x":1060,"y":380,"wires":[["efa29d567fc7a8b3"]]},{"id":"efa29d567fc7a8b3","type":"ui_chart","z":"122012a1fbf7f480","name":"","group":"0b883c9bb34f389c","order":3,"width":"0","height":"0","label":"Chart
Humedad","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","nodata":"Humedad","dot":false,"ymin":"10","ymax":"60","removeOlder":1,"r
emoveOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor":false,"useUTC":false,"colors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728",
"#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferentColor":false,"className":"","x":1440,"y":360,"wires":[[]]},{"id":"1d4527d0d3bd5f29","type":"function","z":
"122012a1fbf7f480","name":"Umbral temperatura","func":"// Umbral de temperatura\nconst temperatureThreshold = 35; // Cambia este valor al umbral deseado\n\n//
Obtén el valor de temperatura\nlet temperature = msg.payload[0].temperature;\n\n// Verifica si la temperatura supera el umbral\nif (temperature >=
temperatureThreshold) {\n    // Establece el mensaje de notificación\n    msg.payload = `¡Alerta! Temperatura alta: ${temperature} °C`;\n    msg.topic = \"Alerta
de Temperatura\"; // Este texto se mostrará en el campo `Topic` del nodo de notificación\n} else {\n    // Si no se alcanza el umbral, detiene la notificación\n
msg = null;\n}\n\nreturn
msg;\n","outputs":1,"timeout":0,"noerr":0,"initialize":"","finalize":"","libs":[],"x":1020,"y":160,"wires":[["915de1e7e86b9a97","33d629d585355f54"]]},{"id":"33d62
9d585355f54","type":"ui_toast","z":"122012a1fbf7f480","position":"top
right","displayTime":"3","highlight":"","sendall":true,"outputs":0,"ok":"OK","cancel":"","raw":false,"className":"","topic":"","name":"","x":1550,"y":80,"wires":[
]},{"id":"4d30507e2313512b","type":"function","z":"122012a1fbf7f480","name":"Umbral humedad","func":"// Umbral de Humedad\nconst humidityThreshold = 30; // Valor
de umbral deseado\n\n// Obtén el valor de Humedad\nlet humidity = msg.payload[0].humidity;\n\n// Verifica si la Humedad supera el umbral\nif (humidity >=
humidityThreshold) {\n    // Establece el mensaje de notificación\n    msg.payload = `¡Alerta! Humedad alta: ${humidity} °C`;\n    msg.topic = \"Alerta de
Humedad\"; // Este texto se mostrará en el campo `Topic` del nodo de notificación\n} else {\n    // Si no se alcanza el umbral, detiene la notificación\n    msg =
null;\n}\n\nreturn
msg;\n","outputs":1,"timeout":0,"noerr":0,"initialize":"","finalize":"","libs":[],"x":1030,"y":460,"wires":[["966163508a05286e","e4b46db033c72b31"]]},{"id":"96616
3508a05286e","type":"ui_toast","z":"122012a1fbf7f480","position":"top
right","displayTime":"5","highlight":"","sendall":true,"outputs":0,"ok":"OK","cancel":"","raw":false,"className":"","topic":"","name":"","x":1550,"y":440,"wires":
[]},{"id":"125ff8c64847fc0e","type":"ui_dropdown","z":"122012a1fbf7f480","name":"","label":"","tooltip":"","place":"Select
option","group":"0b883c9bb34f389c","order":2,"width":0,"height":0,"passthru":true,"multiple":false,"options":[{"label":"10 minutos","value":"10
MINUTE","type":"str"},{"label":"1 hora","value":"1
HOUIR","type":"str"}],"payload":"","topic":"topic","topicType":"msg","className":"","x":1060,"y":560,"wires":[[]]},{"id":"a3a9bcc1eef4525f","type":"MySQLdatabase"
,"name":"iot_data","host":"127.0.0.1","port":"3306","db":"iot_data","tz":"","charset":"UTF8"},{"id":"b7c01e510df0d097","type":"ui_group","name":"Sensores","tab":"
4a6858cc09140a6e","order":1,"disp":true,"width":"6","collapse":false,"className":""},{"id":"0b883c9bb34f389c","type":"ui_group","name":"Chart","tab":"4a6858cc0914
0a6e","order":2,"disp":true,"width":"6","collapse":false,"className":""},{"id":"4a6858cc09140a6e","type":"ui_tab","name":"esp32_tecnicatura","icon":"dashboard","o
rder":1,"disabled":false,"hidden":false}]
```

5. Conclusiones

La implementación de este sistema IoT demuestra la efectividad de una solución escalable y flexible que no solo captura y almacena datos de sensores, sino que también facilita su visualización en tiempo real mediante un dashboard en Node-RED. Esto permite a los usuarios monitorear métricas y tomar decisiones rápidas, demostrando el valor de integrar una capa de visualización en proyectos IoT. La estructura modular asegura que el sistema pueda seguir expandiéndose con más dispositivos y métricas en el futuro.