Diseño e Implementación de Sistemas Embebidos, Laboratorio 2.

Nombre: Joan Esteban Velasco Larrea.

Reto 1 – ESP32 Standalone Web Server

Análisis de Requerimientos.

Objetivo:

Diseñar e implementar un servidor web HTTP autónomo ejecutándose sobre un ESP32, capaz de monitorear variables ambientales (temperatura y humedad) y permitir la interacción del usuario a través de una interfaz web accesible desde cualquier dispositivo dentro de la misma red local. El sistema debe ser no bloqueante, confiable y fácilmente extensible.

Requerimientos Funcionales.

1. Monitoreo Ambiental
   - El sistema debe leer temperatura y humedad relativa usando un sensor DHT22.
   - Las variables deben actualizarse de manera periódica.
   - Los valores deben:
     - Mostrarse en una interfaz web
     - Enviarse por el monitor serial
     - Visualizarse localmente en una pantalla OLED
2. Interfaz de Usuario (Web)
   - El sistema debe proporcionar una página web http alojada directamente en el ESP32.
   - La interfaz debe permitir al usuario:
     - Ingresar el setpoint de velocidad del motor (RPM).
     - Visualizar el valor del RPM.
   - El hardware del motor no es obligatorio, por lo que el enfoque está en la interfaz y el flujo de datos.
   - El setpoint ingresado debe mostrarse en tiempo real en el monitor serial.
3. Requisitos de la Interfaz Web
   - La página debe ser responsive, accesible desde celulares, tablets y computadores.
   - La interfaz debe mostrar claramente:
     - Temperatura
     - Humedad
     - Campo de entrada para RPM

    - Timestamp de la última actualización (hora obtenida vía NTP).

Requerimientos no Funcionales.

1. Conectividad:
   - El sistema debe conectarse a una red WiFi local.
   - El servidor HTTP debe operar en el puerto 80.
2. Confiabilidad:
   - El sistema debe mantener la operación continua del servidor web.
   - La lectura de sensores no debe bloquear la atención de clientes HTTP.
   - El tienestamp debe mantenerse sincronizado usando NTP.
3. Simplicidad y Escalabilidad:
   - La arquitectura debe separar claramente:
     - Adquisición de datos
     - Servidor Web
     - Interfaz de usuario

Análisis y Diseño del sistema (Arquitectura y Componentes).

Arquitectura General.

El sistema está compuesto por los siguientes bloques funcionales:

- Capa de Sensado
  - Sensor DHT22 (Temperatura y Humedad)
- Capa de Procesamiento
  - ESP32 (Control lógico, servidor web y sincronización NTP)
- Capa de Interfaz Local
  - Pantalla OLED SSD1306 (I2C).
- Capa de Comunicación
  - WiFi (HTTP + NTP)
- Capa de Usuario
  - Navegador Web (PC o Móvil).

Componentes Utilizados

- Microcontrolador ESP32
- Sensor DHT22
- Pantalla OLED SSD1306 (128x64, I2C).
- Red WiFi local
- Navegador Web
- Entorno de Desarrollo Arduino IDE.

Diseño e Integración de Hardware.

El diseño de este reto se separó en dos partes, una para la medida de temperatura y humedad (figura 1a) y otra para el control del motor (figura 1b). En la figura 1a el potenciómetro representa un sensor lm35.

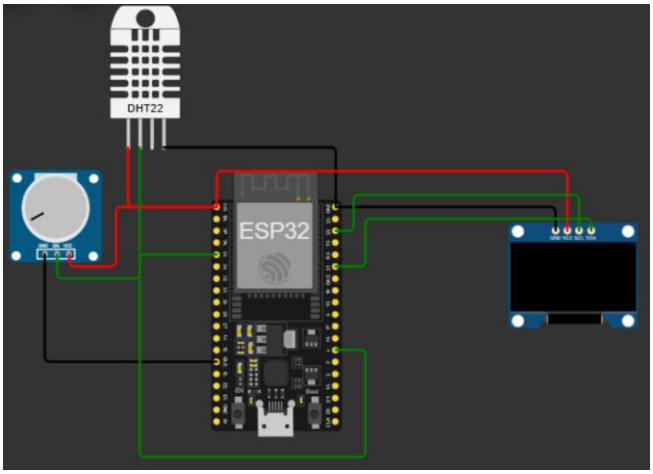

Figura 1a. Arquitectura de hardware utilizada, temperatura y humedad.

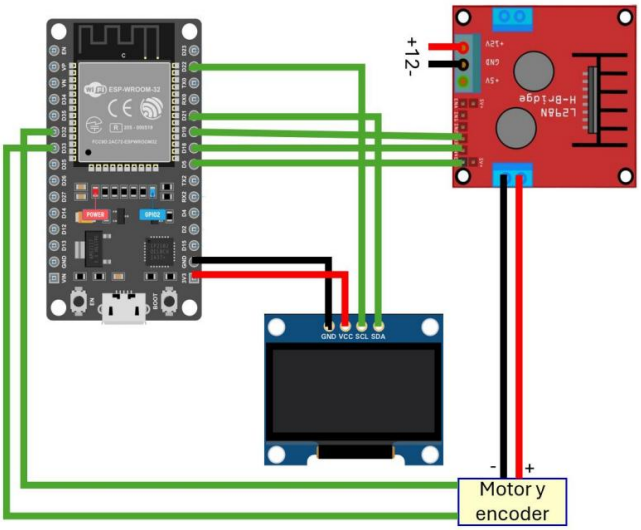

Figura 1b. Arquitectura de hardware utilizada, control motor.

Desarrollo del firmware.

Organización del Código.

El reto se implementa en dos variantes funcionales:

1. ex_1_1.ino – Monitor Ambiental Web
   - Lectura del sensor DHT22.
   - Servidor Web HTTP con endpoint /data.

- Envío de datos en formato JSON.
- Interfaz web que actualiza datos mediante fetch().
- Sincronización de hora NTP para generar timestamps.
- Visualización local en pantalla OLED.

2. ex_1_2.ino – Interfaz Web para Setpoint de Motor
    - Página web con campo de entrada para RPM
    - Envío del setpoint en el monitor serial
    - Arquitectura preparada para integración con control de motor (PID).
    - Uso de NTP para timestamp de actualización.
    - Máquina de estados para control no bloqueante.

Los códigos correspondientes se encuentran en la carpeta ex1 del respositorio en el apartado de referencias.

Interfaces y Protocolos

- Interfaces Hardware
    - I2C
        - OLED SSD1306
        - SDA GPIO 21
        - SCL GPIO 22
    - GPIO
        - DHT22 GPIO15
    - ADC
        - Entrada analógica para temperatura adicional.
- Protocolos de comunicación
    - HTTP
        - / (página principal)
        - /data (datos ambientales)
        - /set_rpm (Recepción de setpoint)
        - /status (estado del sistema)
    - NTP
        - Sincronización horaria para timestamps
    - WiFi
        - Conexión a red local

Validación y Resultados.

- El servidor web es accesible desde cualquier dispositivo conectado a la red.
- Los valores de temperatura y humedad se actualizan periódicamente.
- El timestamp refleja correctamente la hora local sincronizada por NTP.

- El setpoint ingresado desde la web se recibe correctamente y se muestra por Serial.
- El sistema opera de forma estable sin bloqueos.

Se logró implementar exitosamente un **servidor web autónomo en ESP32**, integrando sensado ambiental, visualización local, sincronización temporal y una interfaz web interactiva.
El diseño modular y basado en estados permite escalar el sistema hacia aplicaciones más complejas como control remoto de actuadores reales o integración con plataformas IoT.

Reto 1 – ESP32 MQTT-Based IoT System

Análisis de Requerimientos.

Objetivo.

Diseñar e implementar un sistema IoT basado en el protocolo MQTT, donde un ESP32 publique de manera periódica variables ambientales (temperatura y humedad) y se suscriba a un tópico de control para recibir un setpoint de velocidad de motor (RPM). El sistema debe utilizar un broker MQTT en la nube, emplear mensajes en formato JSON y justificar adecuadamente la selección del nivel de calidad de servicio (QoS).

Requerimientos Funcionales.

1. Publicación de Telemetría
   - El sistema debe medir temperatura y humedad utilizando un sensor DHT22.
   - Los valores deben publicarse de forma periódica mediante MQTT.
   - El tópico de publicación debe seguir una estructura clara y jerárquica:
     - Esp32/sensors/dht22
   - Los mensajes deben enviarse en formato JSON limpio y estructurado.
2. Visualización y Registro de Datos.
   - Los datos publicados deben poder visualizarse en un dashboard MQTT.
   - Los datos publicados por el ESP32 son visualizados en tiempo real mediante un dashboard desarrollado en Node-RED, el cual actúa como cliente MQTT. Node-RED permite tanto la visualización gráfica de la temperatura y humedad a lo largo del tiempo como la interacción del usuario para el envío de consignas de control.
   - La visualización debe permitir observar la evolución temporal de las variables ambientales.
3. Suscripción a Setpoint de Control
   - El sistema debe suscribirse a un tópico de control.
     - Esp32/control/rpm

    - El setpoint de velocidad (RPM) es enviado desde una interfaz gráfica desarrollada en Node-RED, la cual publica el valor en el tópico.
  - El mensaje recibido debe contener el setpoint de RPM en formato JSON.
  - El valor recibido debe mostrarse en tiempo real através del monitor serial
  - El hardware del motor no es obligatorio; el enfoque está en la comunicación y flujo de datos.
4. Conectividad con bróker.
  - El sistema debe conectarse a un bróker MQTT en la nube.
  - Se debe emplear comunicación segura mediante TLS.
  - El bróker utilizado es HiveMQ Cloud.
5. Calidad de servicio (QoS)
  - Se debe seleccionar y justificar el nivel de QoS empleado tanto en publicación como en suscripción.

Requerimientos no Funcionales.

1. Confiabilidad
  - El sistema debe reconectarse automáticamente al bróker MQTT en caso de pérdida de conexión.
  - La comunicación no debe bloquear la ejecución principal del programa.
2. Escalabilidad y Organización.
  - La estructura de tópicos debe permitir la integración futura de nuevos sensores o actuadores.
  - El uso de JSON debe facilitar la interoperabilidad con plataformas IoT externas.
3. Seguridad
  - La conexión con el bróker debe realizarse mediante TLS.
  - Las credenciales deben manejarse de forma explícita en la configuración.

Análisis y diseño del Sistema (Arquitectura y Componentes).

Arquitectura general.

El sistema se organiza en los siguientes bloques funcionales:

- Capa de Sensado:
  - Sensor DHT22 (temperatura y humedad).
- Capa de Procesamiento:
  - ESP32 (lógica de control y comunicación MQTT).
- Capa de Comunicación
  - WiFi
  - Protocolo MQTT sobre TLS.
- Capa de Visualización y Control

    - Dashboard MQTT
    - Interfaz gráfica en Node-RED
    - Monitor Serial

Componentes Utilizados.

- Microcontrolador ESP32.
- Sensor DHT22.
- Red WiFi local.
- Broker MQTT HiveMQ Cloud.
- Dashboard MQTT (HiveMQ Web Client y Node-RED).
- Entorno de desarrollo Arduino IDE.

Diseño e Integración de Hardware.

El diseño de hardware sigue la misma estructura que en el anterior reto (figuras 1a y 1b).

Desarrollo de Firmware.

Organización del Código

El firmware se estructura en los siguientes módulos funcionales:

- Conectividad WiFi:
    - Establece la conexión a la red local.
- Cliente MQTT
    - Configuración de bróker, credenciales y TLS.
    - Publicación y suscripción a tópicos.
- Adquisición de Sensores:
    - Lectura periódica del DHT22.
- Procesamiento de Mensajes:
    - Serialización y deserialización JSON.
- Gestión de Setpoint:
    - Recepción del setpoint de RPM desde MQTT.
    - Visialización del valor recibido por el monitor serial.

Interfaces y Protocolos.

Interfaces de Hardware:

- GPIO
    - Sensor DHT22 (GPIO 4).
- PWM
    - Señal para control del motor.

Protocolos de Comunicación:

- WiFi:
  - Conexión a red local
- MQTT
  - Publicación:
    - Esp32/sensors/dht22
  - Suscripción:
    - Esp32/control/rpm
- TLS
  - Comunicación segura con bróker MQTT.

Formato de Mensajes:

Publicación de Sensores:

```
{
 "temperature": 25.4,
 "humidity": 60.2,
 "unit": {
  "temperature": "C",
  "humidity": "%"
 }
}
```

Setpoint de Control:

```
{
 "rpm": 1200
}
```

Selección y Justificación del QoS.

- Publicación de sensores:
  Se utiliza QoS 0, ya que se trata de datos periódicos de telemetría. La pérdida ocasional de un mensaje no afecta el funcionamiento general del sistema.
- Suscripción al setpoint:
  Se utiliza QoS 1, garantizando que el setpoint de RPM llegue al menos una vez al ESP32. Esto es importante para evitar la pérdida de comandos de control.

Validez y resultados.

- El ESP32 se conecta correctamente al bróker MQTT en la nube.
- Los valores de temperatura y humedad se publican periódicamente.
- Los datos pueden visualizarse correctamente desde el tópico de control.
- El valor recibido se muestra en tiempo real por el monitor serial.
- El sistema opera de manera estable y no bloqueante

Interfaz Gráfica.

Se utilizó la herramienta de Node-RED para la creación de una interfaz gráfica donde observar los resultados de temperatura y humedad, además de modificar el RPM. La siguiente figura muestra el flujo de datos:

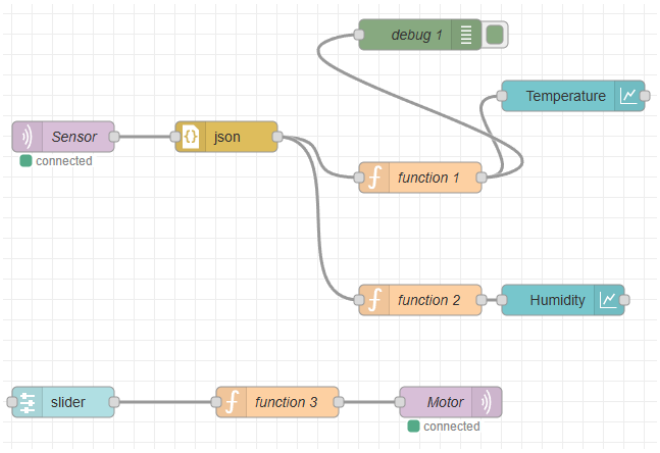

Figura 2. Flujo de datos Node-RED.

En este diagrama se muestra como con Node-RED se suscribe a los tópicos correspondientes a los sensores, y así poder leer y visualizar en un gráfico las medidas obtenidas (figura 3).

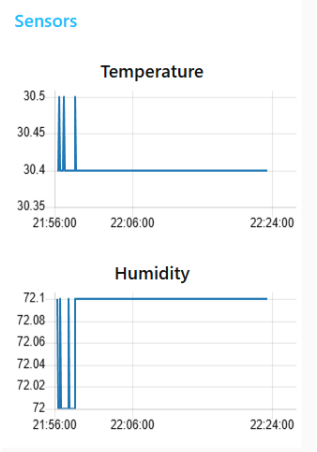

Figura 3. Visualizar datos de sensores.

De igual forma se observa como con Node-RED también se publica en el tópico que modifica el RPM del motor. De esta forma con un slider se pude modificar dicho valor.

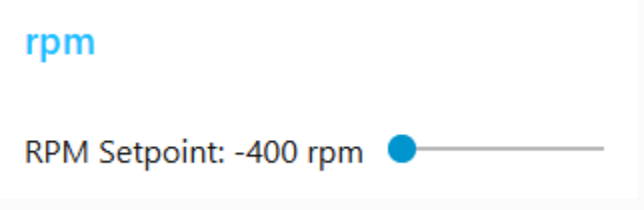

Figura 4. Slider que modifica el RPM de un motor.

Conclusiones.

Se logró implementar exitosamente un sistema IoT basado en MQTT utilizando el ESP32, integrando sensado ambiental, comunicación segura con un broker en la nube y control remoto mediante tópicos de suscripción. El uso de una estructura clara de tópicos, mensajes en formato JSON y niveles de QoS adecuados permite que el sistema sea escalable y fácilmente integrable con plataformas IoT más complejas.

Referencias.

Repositorio Github: https://github.com/joanvel/Lab02_Diseno_e_Implementacion_de_Sistemas_Embebidos/tree/main