Proyecto: Sistema Embebido IoT

Franklin Mauricio Gómez López[0107100596]

franklin.gomez@ucuenca.edu.ec;

Abstract

Este documento presenta el desarrollo de un sistema embebido IoT, enfocado en la simulación de un entorno de monitoreo y control para un vivero y un pequeño estanque. El sistema se basa en un microcontrolador ESP32, simulado junto con diversos sensores y actuadores mediante la plataforma WokWi. La comunicación entre los componentes se realiza a través del protocolo MQTT, utilizando un *broker* Mosquitto configurado localmente.
Los datos recolectados son enviados al *broker* MQTT, el cual interactúa con Node-RED, ejecutado en un dispositivo Android mediante la aplicación Termux. A través de los flujos configurados en Node-RED, se gestionan tanto la visualización de los datos en un *dashboard* interactivo como la publicación de información hacia la plataforma en la nube ThingSpeak, donde es posible observar las variables medidas en tiempo real.

1 Desarrollo

1.1 Descripción del proyecto

El presente proyecto tiene como objetivo el monitoreo de un vivero y un estanque (o pecera), integrando sensores y actuadores en un sistema embebido con conectividad IoT. En el vivero se miden parpametros como la temperatura, la humedad ambiental y el nivel de reserva de agua, además de incluir un sistema de riego controlado. En el caso de la pecera, se monitoriza la temperatura del agua y se gestiona la alimentación de los peces.

La Figura 1 muestra la arquitectura general del sistema. Todos los componentes físicos, incluyendo sensores, actuadores y el microcontrolador, en este caso un ESP32, se simulan localmente mediante la plataforma *Wokwi*, utilizando la extensión para *Visual Studio Code*. El ESP32 obtiene las lecturas de los sensores y publica los datos mediante el protocolo MQTT hacia un *broker* Mosquitto, que se ejecuta en la

misma máquina donde se realiza la simulación. Esta configuración permite facilitar la conexión WiFi simulada del microcontrolador.

Paralelamente, se implementa una interfaz en un teléfono Android mediante la aplicación *Termux*, sobre la cual se ejecuta Node-RED. A través de los nodos MQTT de Node-RED, se gestionan las suscripciones y publicaciones hacia los mismos temas utilizados por el ESP32. La Figura 2 presenta un resumen de los *topics* empleados. Finalmente, Node-RED envía los datos hacia la plataforma ThingSpeak para su visualización en un canal público.

Todo el proyecto se puede encontrar en este repositorio de GitHub, en dicho repositorio también se encuentra el archivo del diagrama de nodos de Node-RED en formato JSON

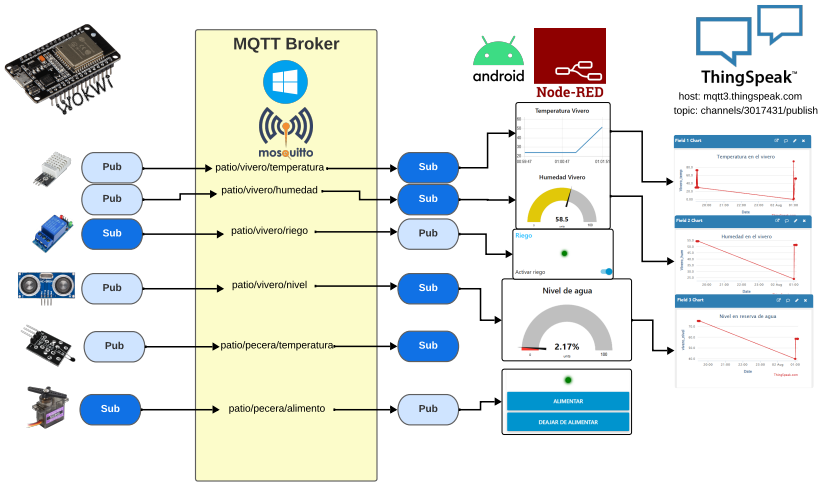

Fig. 1 Arquitectura general del sistema: simulación en Wokwi, comunicación MQTT local y envío a la nube mediante Node-RED

1.2 Simulación en WokWi

Como se mencionó anteriormente, se utilizó un microcontrolador ESP32 para la simulación del sistema. La medición y el control de variables se llevaron a cabo mediante los siguientes sensores y actuadores:

- **DHT22**: Sensor digital empleado para medir la temperatura y la humedad en el vivero.
- **Módulo relé**: Utilizado para activar o desactivar el sistema de riego a través de comandos enviados desde Node-RED.
- **HC-SR04**: Sensor ultrasónico destinado a la medición del nivel de agua en el depósito de riego.

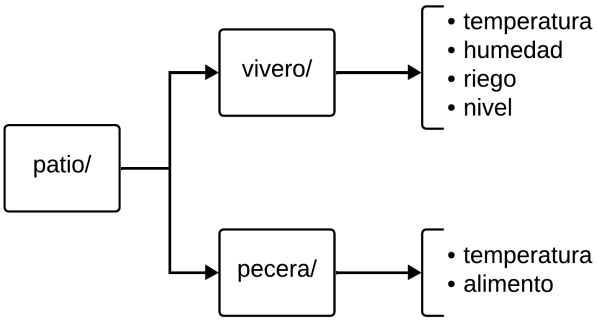

Fig. 2 Resumen de los *topics* MQTT utilizados para la transmisión de datos y control de eventos

- **NTC**: Sensor resistivo de temperatura utilizado para monitorear la temperatura del agua en el estanque.
- **Micro servomotor**: Controla un mecanismo de apertura y cierre encargado de dosificar la alimentación de los peces.

La Figura 3 muestra la disposición de los componentes en el entorno de simulación *WokWi*.

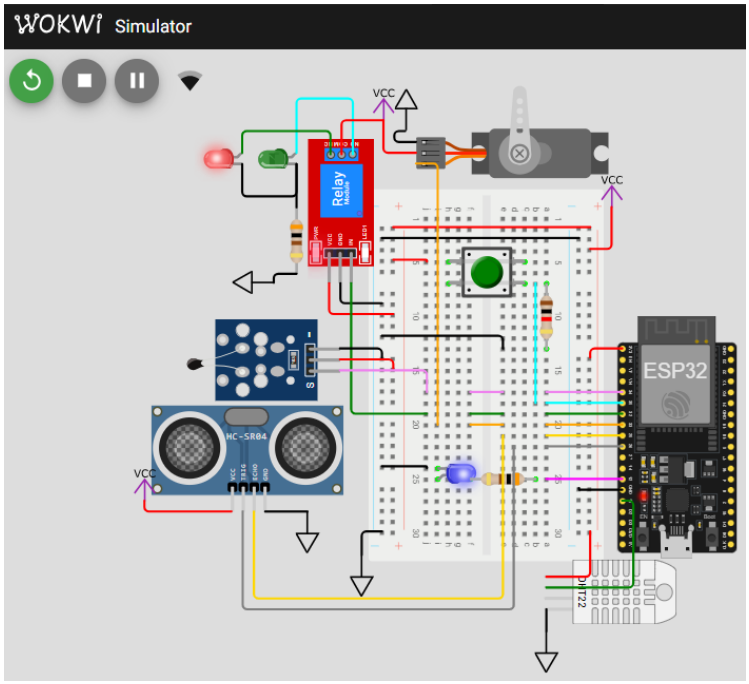

Fig. 3 Simulación del circuito completo en Wokwi, incluyendo sensores, actuadores y el ESP32.

1.3 Ejecución de Node-RED

1.3.1 Instalación de Node-RED en Android

Para ejecutar Node-RED en un dispositivo Android, es necesario contar con una terminal funcional en el sistema operativo. En este proyecto se utilizó la aplicación *Termux*, una terminal de código abierto ampliamente utilizada para entornos tipo Unix en Android. Se recomienda instalar la versión más reciente desde la tienda de aplicaciones *F-Droid*, ya que la versión disponible en Google Play Store ha dejado de recibir actualizaciones.

Una vez instalada la aplicación, se deben ejecutar los siguientes comandos para actualizar los paquetes del sistema:

```
pkg update && pkg upgrade
```

Posteriormente, se instala Node.js, ya que Node-RED depende de este entorno de ejecución para funcionar:

```
pkg install nodejs-lts
```

Una vez instalado Node.js, se procede con la instalación de Node-RED utilizando el gestor de paquetes `npm`:

```
npm install -g --unsafe-perm node-red
```

De manera opcional, puede instalarse un *broker* MQTT local (por ejemplo, Mosquitto). Sin embargo, en este proyecto no se utilizó el *broker* local del dispositivo móvil, ya que el ESP32 simulado no es capaz de establecer conexión con él.

```
pkg install mosquitto
```

Finalmente, para iniciar Node-RED, se ejecuta el siguiente comando:

```
node-red
```

1.3.2 Flujo de trabajo en Node-RED

En la Figura 4 se muestra el flujo de trabajo configurado en Node-RED, junto con los distintos nodos empleados en este proyecto.

Los nodos principales son los de comunicación MQTT: `mqtt in` y `mqtt out`, los cuales permiten la suscripción y publicación en temas, respectivamente. En este proyecto se utilizaron tres nodos de publicación. Dos de ellos están asociados al control de actuadores, el sistema de riego y la alimentación de los peces, mediante la interfaz del *dashboard* de Node-RED. Para esto, se emplearon nodos de tipo `button` y `switch`, los cuales permiten la interacción directa del usuario con el sistema. A través de estos controles, se activa el módulo relé y el servomotor, respectivamente.

El tercer nodo de publicación MQTT se utilizó para enviar los datos recolectados por los sensores a la plataforma ThingSpeak. Dado que ThingSpeak recibe múltiples

campos de datos a través de una única publicación en un solo *topic*, se emplearon nodos de tipo `function` para agrupar y formatear los datos antes de enviarlos, garantizando una estructura compatible con los requerimientos de la plataforma.

Además, el *dashboard* de Node-RED incorpora diversos *widgets* que permiten la visualización de datos mediante gráficas, indicadores y elementos visuales adicionales, como un LED que refleja el estado del sistema. También se incluyó un nodo de notificación que genera una alerta visual en la interfaz cuando el nivel de agua en el depósito es muy bajo, como se observa en la Figura 5.

El flujo de trabajo completo se puede encontrar en este enlace dentro del repositorio del proyecto.

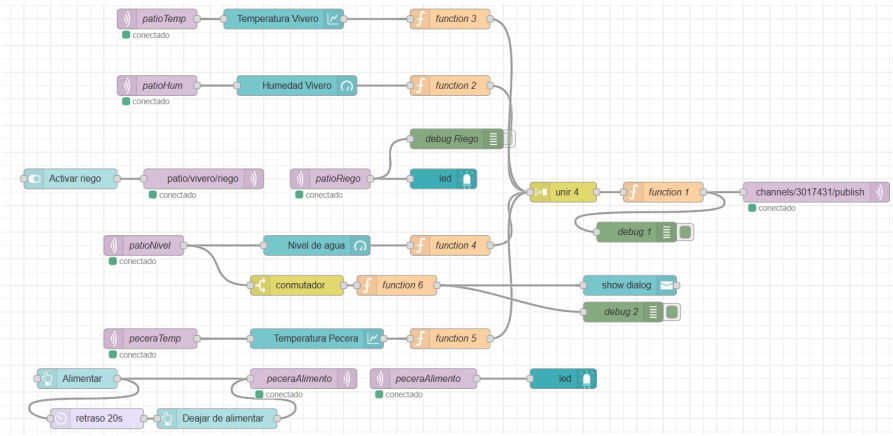

Fig. 4 Flujo de trabajo en Node-RED: nodos de adquisición, procesamiento, control y visualización.

1.4 Gráficas de ThingSpeak

Para la visualización de los datos obtenidos por los sensores, se configuró un canal público en la plataforma ThingSpeak, el cual se encuentra disponible en el siguiente enlace: `thingspeak.mathworks.com/channels/3017431`.

En este canal se habilitaron cuatro campos, correspondientes a las siguientes variables monitorizadas:

- Temperatura del vivero.
- Humedad del vivero.
- Nivel de agua en el depósito de riego.
- Temperatura del agua en la pecera.

La Figura 6 muestra la visualización de estos datos en forma de gráficas de tiempo, generadas automáticamente por la plataforma a partir de los datos enviados mediante MQTT desde Node-RED.

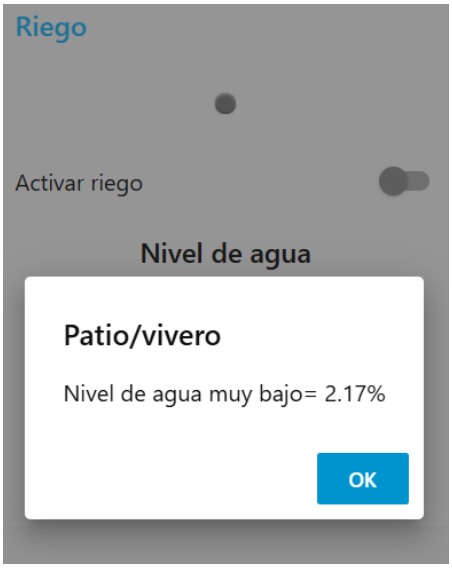

Fig. 5 Alerta visual en la interfaz de Node-RED ante un nivel bajo de agua.

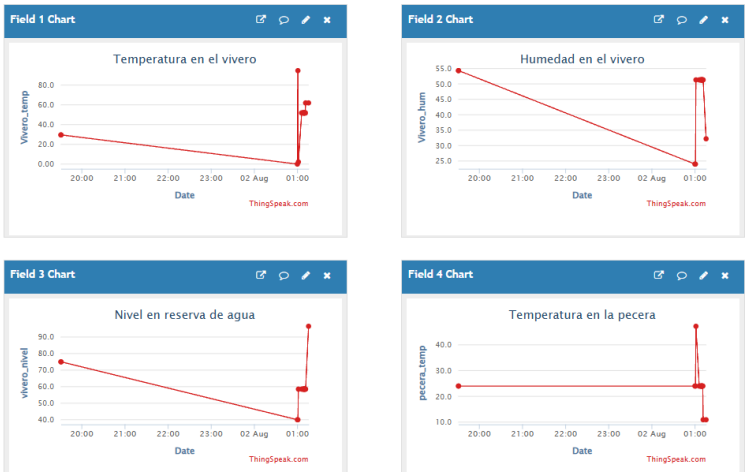

Fig. 6 Visualización en ThingSpeak: gráficas de las variables medidas por el sistema IoT.