Informe del trabajo final

Internet de las Cosas

Nicolás Caporal - 21322/4

Sistema de Riego Automatizado IoT para Planta en Maceta

1. Diseño

El sistema se articula alrededor de un microcontrolador ESP32 que simula los dos sensores principales del invernadero: la humedad del suelo y el nivel de agua de un tanque. La lectura de humedad se genera mediante una función senoidal en software, configurable en periodo y amplitud para emular las fluctuaciones naturales del terreno. Cuando la humedad simulada cae por debajo del umbral establecido (30 %), el ESP32 activa el LED integrado —equivalente a encender una bomba de riego— y decrece el porcentaje de agua disponible en el tanque. Al llegar a 0 % se inicia un temporizador interno de 30 s antes de reponer el nivel al 100 %, para simular una acción no inmediata por parte de un humano, y poder verificar el correcto funcionamiento con el tanque vacío del presente prototipo durante las pruebas.

La comunicación entre el ESP32 y el servidor se realiza mediante MQTT, publicando cada 5 s en dos topics: invernadero/humedad e invernadero/nivel. Un broker Mosquitto local gestiona las conexiones TCP/IP. Posteriormente, Node-RED suscribe ambos topics y usa un nodo “join” para sincronizar humedad y nivel en un solo mensaje, que un nodo “function” formatea con los campos y tags requeridos. Este mensaje compuesto se envía al nodo de salida influxdb out, donde queda almacenado en la base de datos InfluxDB 1.8 bajo la measurement “invernadero” con los campos humedad y nivel y la etiqueta maceta=1.

Para la visualización y el análisis, Grafana consume los datos de InfluxDB y presenta dos paneles de tipo “Time series”. El panel de humedad muestra el promedio de valores agrupados cada segundo, donde se aprecia claramente la onda senoidal, y cuenta con un threshold en el valor de 30, representando el umbral de humedad y el momento donde se empieza a regar. El panel de nivel utiliza la última lectura disponible y dispone una regla de alerta que dispara cuando el nivel cae por debajo de 20 %, evaluada cada minuto y con un periodo “for” de 30 s. Cuando se cumple la condición, Grafana envía un webhook al bot de Telegram configurado con BotFather, notificando al usuario que es momento de rellenar el tanque.

2. Justificación técnica

Seleccionamos el ESP32 para la simulación debido a su bajo coste, su amplio soporte de librerías para Wi-Fi y MQTT, y su facilidad para escalar en el futuro hacia sensores físicos sin modificar el resto de la arquitectura. Generar la humedad con una función senoidal en lugar de valores aleatorios proporcionó datos predecibles y repetibles, esenciales para verificar con precisión la lógica de riego y el disparo de las alertas en Grafana. Además, basarnos en millis() en lugar de delay() evitó bloqueos en el bucle principal y garantizó publicaciones MQTT regulares cada cinco segundos, manteniendo la reconexión automática del cliente cuando fuera necesario.

Decidí trasladar al servidor —mediante Node-RED— toda la lógica de agrupación, formateo y persistencia de datos en lugar de implementarla en el ESP32 para desacoplar responsabilidades y mantener el firmware del microcontrolador lo más simple posible. Gracias a los nodos “mqtt in”, “join” y “function”, cualquier cambio en la estructura del mensaje, la incorporación de nuevos tags o la aplicación de filtros se realiza de forma visual y sin tocar el código desplegado. De este modo, el ESP32 sólo publica valores crudos, mientras que Node-RED se encarga de gestionar errores, reintentos de conexión y enriquecer los datos antes de enviarlos a InfluxDB.

Por último, la combinación de InfluxDB y Grafana ofrece un flujo de trabajo optimizado para series temporales: InfluxDB se ocupa de las escrituras frecuentes con alta eficiencia y Grafana proporciona paneles personalizables, zoom temporal y un sistema de alertas unificado. La notificación a Telegram vía webhook cierra el ciclo de manera centralizada, sin añadir complejidad al microcontrolador y permitiendo, en el futuro, cambiar de canal de notificación (por ejemplo a Slack o correo) sin desplegar nuevo firmware.

3. Resultados

En las pruebas realizadas, el sistema demostró estabilidad y precisión. El panel de humedad en Grafana refleja con nitidez la curva senoidal, lo que valida la generación programada de datos. La lógica de consumo y recarga de tanque se comporta según lo esperado: al cruzar el umbral de humedad, el LED integrado se enciende, el nivel decrementa en incrementos de 10 %, y tras llegar a 0 % el tanque permanece vacío durante 30 s antes de volver automáticamente al 100 %. Estas etapas son claramente visibles en el panel de nivel y coinciden con los logs seriales y lo programado en el ESP32.

La regla de alerta en Grafana se activó correctamente cada vez que el nivel descendió por debajo de 20 %, así como también se marca correctamente de manera gráfica cuando el nivel del agua se recupera. Las notificaciones llegaron en menos de 30 s al chat de Telegram configurado, lo que permitiría al usuario reaccionar con celeridad. Durante ejecuciones continuas de varias horas no se registraron desconexiones ni pérdidas de mensaje MQTT.

4. Pendiente a futuro

Quedan pendientes varias mejoras para llevar este prototipo al entorno real. En primer lugar, habría que considerar la opción de incorporar Deep Sleep en el ESP32 para reducir drásticamente el consumo y permitir un despliegue con baterías sin depender de una fuente de energía constante. En paralelo, será imprescindible sustituir la simulación por sensores físicos —sensor de humedad del suelo, ultrasonido para nivel y una bomba real— y ajustar empíricamente parámetros como la duración de activación de la bomba o el umbral de humedad según las características de cada planta; esas pruebas de campo quedan fuera del alcance de este trabajo, pero son clave para definir valores de operación reales. Gracias a la arquitectura modular implementada —donde toda la lógica de simulación está perfectamente encapsulada en funciones independientes, y la lógica de persistencia, notificación y visualización están del lado del servidor—, bastaría con reemplazar esas partes del código por lecturas de hardware real para que el resto del sistema continúe funcionando sin modificaciones.