Proyecto IoT para Informática Industrial

Presentación y contexto

Planteamos una solución completa en el ámbito de "Internet de las cosas" (IoT) diseñando un dispositivo prototipo usando una placa de desarrollo basada en el módulo WiFi ESP32-C3 programada con el IDE de Arduino. Este dispositivo será capaz de leer sensores, enviar información y recibir órdenes para cambiar su configuración y el estado de los actuadores que tiene conectados mediante protocolos de comunicación apropiados para IoT. Diseñaremos una aplicación que controlará una serie de dispositivos asociados (al menos uno por cada integrante del grupo de trabajo) que se encontrarán distribuidos geográficamente formando una red de sensores y actuadores. Los datos (medidas de los sensores, estado de los actuadores, etc.) generados por los dispositivos serán recogidos por una aplicación que los procesa y almacena. La información de interés será accesible para los usuarios de una forma amigable utilizando una variedad de interfaces, al menos un interfaz gráfico web y otro conversacional Telegram.

Estamos desarrollando un sistema completo con la intención de que sea de la máxima utilidad, por ello, es fundamental generar un sistema bien estructurado y con una documentación completa y exhaustiva, tanto a nivel de usuario como de desarrollo, que facilite su explotación, mantenimiento y ampliación. El funcionamiento global del sistema debe ser lo más fiable y eficiente posible.

El proyecto realizado debe cumplir el planteamiento y propuesta de requisitos descritos a continuación. Si un grupo de trabajo quiere proponer un proyecto diferente, usar sensores distintos, o cualquier otra modificación, puede hacerlo planteando previamente la idea a los profesores de la asignatura.

Descripción y requerimientos del sistema

El proyecto consistirá en diseñar e implementar un sistema de monitorización y control multi-estancia o multi-ubicación que incluirá los siguientes elementos:

Dispositivos IoT

- Disponemos de varios nodos IoT basados en el módulo Wi-Fi ESP32-C3 que se encargarán de medir, almacenar de forma temporal y transmitir de forma inalámbrica, los parámetros fundamentales a medir en cada estancia/ubicación (temperatura, humedad, etc.). También debe funcionar como controlador de dos actuadores a través de dos GPIOs. Uno de ellos controlará una luminaria RGB (a través del GPIO30 / RGB_BUILTIN) y el otro se comportará como interruptor de dos estados encendido/apagado (GPIO5). Los dispositivos recibirán órdenes de forma inalámbrica a través de la red de comunicaciones para cambiar su configuración, cambiar la intensidad y color del LED RGB, mientras que la segunda sólo podrá apagarse o encenderse (GPIO5). También será posible actuar localmente sobre el dispositivo pulsando el micro-botón BOOT conectado al GPIO9. Este GPIO9 estará a nivel alto (HIGH/3.3V), pero cuando se pulsa el botón se pone a nivel bajo (0V). Para que el botón se pueda leer correctamente hay que inicializar el pin GPIO9 en el modo INPUT_PULLUP. Hay un ejemplo de configuración del botón en el CV.

El dispositivo informará de los cambios de estado que se experimenten en los circuitos que controla (GPIO5 y LED RGB) de forma inmediata. El envío de los mensajes con los datos de sensores y estado de conexión del dispositivo se harán regularmente a intervalos, inicialmente, de 5 minutos. Este intervalo se podrá cambiar enviando un mensaje MQTT de configuración al dispositivo. Si no se dispone de LED conectado al GPIO5 esa salida seguirá funcionando igualmente pero sin la indicación visual.

- Cuando se produzca un cambio de intensidad o color en el LED RGB, este debe realizarse de forma gradual a una velocidad de ±1% cada 10ms hasta alcanzar la nueva intensidad establecida. Esta velocidad puede ser configurada en el dispositivo mediante mensaje MQTT.
- Por defecto en la salida GPIO5 encendido será nivel alto (HIGH), pero este funcionamiento será configurable mediante mensaje MQTT permitiendo usar la lógica inversa siendo encendido un nivel bajo de salida (LOW). Esto permitirá controlar circuitos externos que trabajen con una u otra lógica de activación.
- Será posible interactuar con el dispositivo a través del botón (BOOT) de la placa conectado al GPIO9. Al **pulsar** el botón, el LED alternará entre apagado o encendido al estado establecido previamente. Al realizar una **pulsación doble** el LED RGB se encenderá a nivel máximo conservando el color programado. Si se pulsa el botón de forma **prolongada** el dispositivo comprobará si hay una actualización disponible (FOTA) en ese momento. Se debe añadir una estrategia software para eliminar posibles rebotes (falsas pulsaciones) en el funcionamiento del botón.
- Los programas de los dispositivos serán actualizables inalámbricamente (FOTA), pudiendo actualizar todos los programas simultáneamente con la misma versión/fichero de firmware si fuese necesario. La actualización se puede programar de tres maneras diferentes:
    - al arrancar el dispositivo
    - cuando se ordene a distancia mediante mensaje MQTT o localmente mediante pulsación de un botón
    - se comprobará periódicamente, cada X minutos. Inicialmente el periodo de comprobación de actualizaciones será 0, lo que significa que no se realizan comprobaciones periódicas de actualización, aunque este parámetro será configurable mediante mensaje MQTT.
- Los nodos se comunicarán con la aplicación en NodeRED mediante un interfaz común usando mensajes MQTT con contenido formateado en JSON. Recordar que el dispositivo debe conectarse al broker MQTT con un **identificador de cliente único** (los números de serie de chip ESP son identificadores únicos). El detalle de los mensajes que deben programarse como mínimo está en la sección “Mensajes MQTT”.

Aplicación de control y supervisión

- Utilizaremos NodeRED para implementar un sistema SCADA (supervisión, control y adquisición de datos) mediante el paquete dashboard. Nuestra aplicación recibirá la

información de los nodos, se encargará de almacenarlos y mostrarlos al usuario permitiendo el control de los actuadores disponibles.

- El sistema maneja múltiples dispositivos simultáneamente de forma que mostrará la información individual de cada uno de ellos, permitirá actuar de forma individual sobre el estado de los actuadores de cada uno, pero también implementará una forma de actuación en grupo que permita cambiar el estado de la iluminación o encendido/apagado, o configuración del dispositivo mediante una única orden del usuario que afectará a todos los dispositivos conectados.

- Mostrará mediante un interfaz gráfico los últimos valores recibidos y también los almacenados, permitiendo consultas por rango de fechas, para obtener resultados clasificados y ordenados y poder representarlos gráficamente. Se realizarán análisis de los datos consultados (valores medios, desviaciones, máximos, etc.). También se podrán **descargar los datos en bruto en un fichero** para su análisis con otras herramientas. El sistema permitirá también visualizar el log (historial) del sistema dónde se almacenan los cambios de estado en los actuadores/LEDs, interacciones de usuario relevantes y errores o avisos que se produzcan en el sistema.

- Como protocolo de comunicaciones IoT hemos seleccionado MQTT. Un broker MQTT (mosquitto) está instalado en iot.ac.uma.es. Este servidor tiene una IP pública por lo que será accesible desde cualquier dispositivo conectado a Internet. De esta forma nuestra aplicación podrá integrar dispositivos que no están en la misma ubicación / LAN a través del servidor común. Se proporcionará un **usuario/contraseña diferente a cada grupo**[1] para poder usar topics independientes en el servidor. Los puertos donde responden los servicios instalados son los estándares:

Servicio	Dirección URL (y puerto)	ejemplo de usuario y clave[1]
mosquitto (MQTT broker)	iot.ac.uma.es:1883	**II99** / **sfd76fsd**[1] (topics: **II99**/#)
mongoDB	iot.ac.uma.es:27017	**II99** / **sfd76fsd**[1] (base de datos: **II99**)
Actualizaciones OTA	https://iot.ac.uma.es:1880/esp8266-ota	

Tabla 1. Servicios disponibles

Base de datos

Usaremos una base de datos para almacenar los datos medidos y poder aceptar peticiones para recuperar y posteriormente analizar/visualizar estos datos. Hay que recordar añadir una fecha a los

[1] consultar **usuario y contraseña** del grupo en los "Comentarios de retroalimentación del profesor" de la tarea "Credenciales del grupo para.." en el CV. Usuario y contraseña será la misma para MQTT y mongoDB. Los topics MQTT deberán comenzar por el nombre de usuario del grupo. El nombre de la base de datos coincide con el nombre de usuario.

datos recibidos antes de almacenarlos. Se ha instalado en iot.ac.uma.es la base de datos NoSQL mongoDB que permite el almacenamiento sencillo y directo de documentos en formato JSON.

Cada grupo de trabajo recibirá un usuario y contraseña para acceder a su base de datos privada en el servidor mongoDB. Todos podemos acceder a la base de datos "ejemplo" para realizar pruebas (ver tabla 1 para dirección y autenticación), hay que tomar la precaución de establecer un nombre de colección único para no interferir con otros grupos cuando utilicemos la base de datos compartida "ejemplo".

La organización en colecciones de los datos gestionados por la aplicación será diseñada a criterio del grupo.

Será necesario **añadir la fecha a los datos almacenados** para poder recuperarlos usando ese criterio.

La base de datos se usará para almacenar como mínimo: los datos recibidos de los sensores y estado de los actuadores y conexión del dispositivo, un historial (log) de los cambios de estado de los actuadores y los posibles errores o avisos que puedan surgir. Para los cambios de estado de los actuadores se registrará el interfaz desde donde se realizó la acción (MQTT/botón local).

Requerimientos no funcionales

- **Corrección y robustez**. El sistema debe producir resultados correctos y adecuados ante las peticiones e interacción del usuario. También debe ser tolerante a errores que puedan producirse en el origen de los datos, algún servicio que no esté disponible o peticiones no coherentes del usuario.
- **Amigabilidad.** El usuario final del sistema podría no tener conocimientos de electrónica o informática, por lo que el interfaz de usuario, operaciones rutinarias de mantenimiento y uso deben ser muy simples y estar perfectamente explicados en un **manual de usuario**.
- **Mantenimiento.** El sistema debe ser fácil de mantener y ampliar. Por ello será necesario que:
    - Nuestros programas y flujos estén bien **organizados y comentados**
    - Facilitemos la documentación adecuada. Todos los elementos del sistema deben estar perfectamente documentados.

Interfaces

Será obligatorio la programación en NodeRED de al menos dos interfaces de usuario:

- Interfaz web gráfica para la interacción con el usuario que se programará mediante el paquete dashboard.
- Interfaz Telegram, que permitirá interactuar con el sistema, solicitar información, cambiar el estado de los actuadores, y usaremos como canal de notificación.
- Aparte también será posible interactuar con el dispositivo a través del botón (BOOT) de la placa conectado al GPIO9.

Por si algún grupo quiere explorar el uso de otras interfaces de usuario que les parezcan interesantes, como los asistentes de voz (Amazon, Google), dejaremos algún ejemplo en el CV.

Dashboard

Esta será la interfaz que ofrezca el mayor número de opciones y funcionalidades. Para ello la interfaz se organizará en distintos “tabs”, o páginas, donde organizar la información y controles. El dashboard tiene nodos que permiten cambiar de página o “tab” lo que puede ser útil para cambiar fácilmente entre las diferentes secciones del interfaz de usuario utilizando botones u otro tipo de controles.

Entre las opciones que permitirá este interfaz podemos contar:

- Mostrar datos actuales (últimas lecturas recibidas) de los sensores y evolución histórica en los últimos días. Incluyendo la hora y fecha de última actualización.
- Recuperar datos de sensores de la base de datos y mostrarlos gráficamente en pantalla. Se podrá seleccionar el rango de fechas de los valores a consultar y se ofrecerán opciones rápidas para el último día, semana, mes, etc.
- Los datos recuperados por consultas descritas anteriormente podrán descargarse a un fichero en formato CSV para su posterior procesamiento.
- Se permitirá el control de los actuadores/LEDs (intensidad de iluminación, encendido y apagado) y se mostrará el último estado conocido de cada uno, junto a la hora/fecha de actualización.
- Se podrá consultar el histórico de cambios de estado de los LEDs registrados en la base de datos.
- El usuario podrá establecer con controles en el dashboard valores máximos y mínimos de alarma para al menos los parámetros de temperatura ambiente y humedad. Si los parámetros alcanzan valores por encima de los máximos o por debajo de los mínimos se notificará a los usuarios a través del canal de Telegram y en el dashboard mediante notificación emergente. También se registrará el evento en el log del sistema (base de datos.

Telegram

Se podrá entablar conversación con el Bot Telegram de nuestro sistema y se podrá preguntar por el estado actual de los actuadores, los últimos valores registrados por los sensores y también nos permitirá cambiar el estado de los actuadores… activando y desactivando los LEDs y cambiando la intensidad del que lo permite, etc. Esta interacción se puede programar con comandos Telegram o conversación, como elija el grupo de trabajo.

El Bot estará implementado en Node-RED haciendo uso de alguno de los paquetes disponibles (node-red-contrib-chatbot, …).

Cada grupo habilitará un canal de difusión Telegram para notificar a los usuarios con las novedades, alarmas y eventos que se puedan producir en el sistema.

Mensajes MQTT

Usar los topics que se indican y los nombres de campos tal y como aparecen en los ejemplos. Sustituir en `IIX` la X por el número de grupo y en `esp32c3-XXXX` poner el identificador obtenido por la función `WiFi.getHostname()` en la placa. Para poder usar los topics que comienzan por `IIX/#` en el broker iot.ac.uma.es es necesario usar el usuario y contraseña del grupo. Si se usa el usuario infind, el topic debe empezar por infind/#. En algunos mensajes se ha añadido un identificador del dispositivo ("CHIPID" basado en el valor `WiFi.getHostname()`) para facilitar su tratamiento y poder diferenciarlo si es necesario.

Topic: IIX/esp32c3-XXXX/conexion
Acción: Publicar
Descripción: Estado de conexión del dispositivo (mensaje retenido). Usar últimas voluntades (LWT) para enviar la notificación desconexión
Ejemplos: {"CHIPID":"esp32c3-XXXX","online":true} {"CHIPID":"esp32c3-XXXX","online":false}

Topic: IIX/esp32c3-XXXX/datos
Acción: Publicar
Descripción: Actualización periódica (5 minutos por defecto) de información de sensores y más.

Ejemplo:

{
"CHIPID":"esp32c3-XXXX",
"Uptime":12300,
"DHT11":{"temp":26.0,"hum":37.0},
"Wifi":{"SSId":"infind","IP":"192.168.0.100","RSSI":-56}
}

Topic: IIX/esp32c3-XXXX/config
Acción: Suscribir
Descripción: Configuración del dispositivo. Se podrá configurar el periodo de envío de mensajes (en segundos), el periodo de comprobación de actualizaciones en minutos y la velocidad de cambio de la salida RGB (milisegundos en cambiar un ±1%). Si el periodo de actualizaciones es 0, no se comprobarán periódicamente. El campo "SWITCH" configura si la salida está activada/encendida a nivel alto (1) o bajo (0).

Si alguno de estos campos tiene valor *null* se mantendrá la configuración previa de ese parámetro.
Ejemplo:

{"envia":300, "actualiza":60, "velocidad":10,"SWITCH":0}
{"envia":600, "actualiza":null, "velocidad":null,"SWITCH":null}

Topic: IIX/esp32c3-XXXX/led/brillo
Acción: Suscribir
Descripción: El dispositivo recibe la configuración de intensidad del LED [0-100]
Ejemplo: { "level":75 }
Es posible añadir un identificador (id) para poder casar la respuesta obtenida en IIX/esp32c3-XXXX/led/brillo/estado con nuestra orden
Ejemplo: { "level":75 , "id":"123456789"}

Topic: IIX/esp32c3-XXXX/led/brillo/estado
Acción: Publicar
Descripción: El dispositivo confirma la configuración de intensidad del LED [0-100] después de aplicarla. Se manda un mensaje único con la intensidad final configurada, no varios mensajes con intensidades intermedias. Se especifica en el mensaje el origen de la orden (pulsador ó mqtt)
Ejemplo: {"CHIPID":"esp32c3-XXXX", "LED":100, "origen":"pulsador"}
Si la orden recibida por mqtt contenía el campo "id" este se copia en el mensaje de cambio de estado
Ejemplo: {"CHIPID":"esp32c3-XXXX", "LED":100, "origen":"mqtt", "id":"123456789"}

Topic: IIX/esp32c3-XXXX/led/color
Acción: Suscribir
Descripción: El dispositivo recibe la configuración de color RGB del LED [0-255]x3
Ejemplo: { "R":255, "G":0, "B":0 }
Es posible añadir un identificador (id) para poder casar la
respuesta obtenida en IIX/esp32c3-XXXX/led/color/estado con nuestra orden
Ejemplo: { "R":255, "G":0, "B":0 , "id":"123456789"}

Topic: IIX/esp32c3-XXXX/led/color/estado
Acción: Publicar
Descripción: El dispositivo confirma la configuración de color RGB del LED después de aplicarla. Se manda un mensaje único con el color configurado.
Ejemplo: {"CHIPID":"esp32c3-XXXX", "R":255, "G":0, "B":0}
Si la orden recibida por mqtt contenía el campo "id" este se copia en el mensaje de cambio de estado
Ejemplo: {"CHIPID":"esp32c3-XXXX", "R":255, "G":0, "B":0, "id":"123456789"}

Topic: IIX/esp32c3-XXXX/switch/cmd
Acción: Suscribir
Descripción: El dispositivo recibe la configuración del interruptor, 1 / 0
Ejemplo: { "level":1 } { "level":0 }
Es posible añadir un identificador (id) para poder casar la
respuesta obtenida en IIX/ESPX/switch/status con nuestra orden
Ejemplo: { "level":1 , "id":"123456789"}

Topic: IIX/esp32c3-XXXX/switch/status
Acción: Publicar
Descripción: El dispositivo confirma la configuración del interruptor después de aplicarla
Ejemplo: {"CHIPID":"esp32c3-XXXX", "SWITCH":1 }
Si la orden recibida por mqtt contenía el campo "id" este se copia en el mensaje de cambio de estado
Ejemplo: {"CHIPID":"esp32c3-XXXX", "SWITCH":1, "id":"123456789"}

Topic: IIX/esp32c3-XXXX/FOTA
Acción: Suscribir
Descripción: El dispositivo recibe la orden de comprobar si hay actualización. El contenido del mensaje no es relevante.
Ejemplo : {"actualiza":true}

Se deben implementar las operaciones sobre los dispositivos (configuración, control de led/switch, actualización) de forma que se permita ordenar estas de forma individualizada o grupal para todos los dispositivos conectados.

Es posible usar comodines para recibir mensajes de un grupo de dispositivos. Pero para enviar mensajes a múltiples dispositivos no es posible. Habrá que, o bien crear nuevos topics para enviar mensajes a todos los dispositivos simultáneamente (por ejemplo, IIX/all/config), o será necesario tener definido el grupo (lista de IDs de los dispositivos) y usarlo para replicar los mensajes a enviar a todos los elementos del grupo.

Ampliaciones

Se puede ampliar/mejorar/modificar el proyecto con las aportaciones que estime del grupo de trabajo. Consulta tus propuestas a los profesores para obtener el visto bueno y posibles recomendaciones. Por ejemplo:

- Utilizar un API-REST para comunicación de los dispositivos con el servidor (alternativa a MQTT) o para consumir algún servicio externo en la nube.
- Utilizar programación de tareas FreeRTOS para implementar el código de los dispositivos.
- Utilizar en algunos dispositivos el protocolo ESP-NOW, utilizando una pasarela ESP-NOW->MQTT como la que tenemos en el laboratorio (código disponible en CV). Se podría también añadir comunicación en la pasarela en la dirección MQTT->ESP-NOW para poder enviar mensajes a los dispositivos ESP-NOW usando MQTT. Por ejemplo para indicarles que hagan una actualización FOTA, o enviarles cualquier otro comando o configuración que sea pertinente en vuestra aplicación. Comparar el funcionamiento de los nodos ESP-NOW y los que usen conexión WiFi.
- Guardar la configuración recibida en la memoria flash del dispositivo de forma que los cambios sean persistentes (librería EEPROM).
- Que el dispositivo permita la configuración de la WiFi la primera vez que se enciende. Si no tiene la WiFi configurada arrancará en modo punto de acceso permitiendo la configuración a través de una página web (hay librerías disponibles para esto: WiFiManager o AutoConnect). Se podría asignar una combinación de pulsaciones en el botón para resetear la WiFi y entrar en modo configuración y poner un nuevo SSID y password.
- Incluir sensores/actuadores adicionales. Se puede utilizar el simulador WOKWI para añadir nuevo HW al proyecto sin necesidad de adquirirlo físicamente.
- …

Entrega:

Se debe entregar un documento PDF con la memoria del proyecto en que se detalle:

- Introducción y objetivos (incluido las funcionalidades implementadas).
- Diseño HW y esquema de conexionado
    - Pines usados para cada sensor/dispositivo
- Diseño SW
    - Diagrama de bloques/flujo del programa
    - Descripción de funcionamiento

    - Técnicas que se han utilizado para hacer más robusto el sistema. Por ejemplo, cómo se maneja el fallo de un sensor, cómo se asegura el paso de mensajes, etc.
    - Librerías utilizadas
- Resultados y Conclusiones
- Manual de usuario
- Lista y descripción de los ficheros entregados
    - Proyectos Arduino (*.ino)
    - Flujos Node-RED (*.json)
    - Proyectos WOKWI si se han utilizado (*.zip)
    - ...