ADMINISTRACIÓN
Y NEGOCIOS

inacap

CloudLed y Android Studio

Evaluación N°4

NOMBRE: Moises Bascur, Ricardo Fernández
CARRERA: Analista Programador
ASIGNATURA: Aplicaciones Móviles IoT
PROFESOR: Manuel Alfredo Sanchez Carcamo
FECHA: 19/12/2025

ADMINISTRACIÓN
Y NEGOCIOS

Inacap

1 Introducción

En el contexto actual de la transformación digital, el Internet de las Cosas (IoT) ha redefinido la manera en que interactuamos con el entorno físico, permitiendo el control y monitoreo de dispositivos desde cualquier lugar del mundo. La capacidad de integrar aplicaciones móviles con microcontroladores mediante servicios en la nube es una competencia fundamental en el desarrollo de soluciones tecnológicas modernas.

El presente informe detalla el desarrollo e implementación del proyecto "CloudLed", un sistema IoT diseñado para establecer una comunicación bidireccional en tiempo real entre una aplicación Android y un microcontrolador ESP32. El propósito central es lograr la sincronización instantánea de estados entre una interfaz de usuario móvil y un actuador físico (LED), utilizando Firebase Realtime Database como el nexo de comunicación.

A lo largo del documento se describen las etapas críticas del proyecto: desde la optimización del hardware utilizando la configuración de resistencia interna INPUT_PULLUP en el ESP32 para reducir componentes externos, hasta la integración de servicios de Google en Android Studio mediante el archivo de configuración google-services.json. Asimismo, se aborda la lógica de programación necesaria tanto en el firmware como en la aplicación móvil para garantizar la consistencia de los datos y la resiliencia del sistema ante reinicios, validando finalmente el funcionamiento mediante pruebas de campo.

2 Objetivos

El objetivo actual es el de integrar una aplicación móvil a un android denominada CloudLed la cual con un microcontrolador fisico ESP32 mediante a firebase Realtime Database para poder lograr la sincronizacion bidireccional de datos y el control de actuadores en tiempo real.

Para algo mas especifico debemos configurar la electrónica del microcontrolador optimizando el hardware implementando un circuito físico utilizando la resistencia interna input_pullup del esp32 para la lectura del botón, asegurando estados lógicos estables HIGH/LOW y eliminando la resistencia externa en la protoboard.

también vamos a implementar una lógica de sincronización y sondeo programando con el firmware del eps32 para que realice consultas periódicas a la base de datos detectando cambios remotos desde la App, ejecutar una rutina de inicio que sincronice el estado del Led físico con el valor almacenado en la nube inmediatamente al encender el dispositivo.

Establecer comunicación bidireccional en tiempo real configurando la base de datos firebase y la aplicación android para que cualquier cambio físico se refleja instantáneamente en la interfaz de usuario , y viceversa, garantizando la consistencia de los datos en la esquema /prueba _iot/estado.

y por último validar el funcionamiento del sistema integrado realizando pruebas de campo que demuestren la capacidad del sistema para recuperarse su estado ante reinicios y mantener la operatividad del LED y la aplicación bajo condiciones de uso normal.

3 Desarrollo

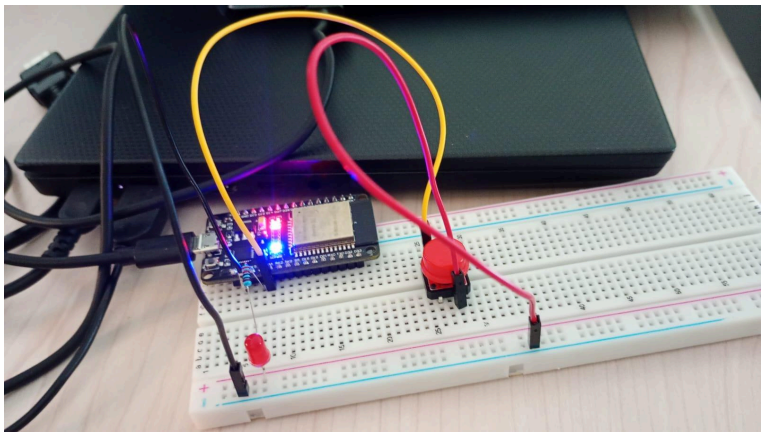
Para la implementación del nodo IoT se seleccionó una placa de desarrollo basada en el chip **ESP32**. A diferencia de alternativas como Arduino Uno, el ESP32 integra conectividad Wi-Fi nativa y Bluetooth en un solo SoC (System on Chip), lo cual es un requisito indispensable para la comunicación directa con la base de datos Firebase RTDB sin necesidad de módulos externos o "shields" adicionales. Además, su lógica de funcionamiento a 3.3V permite un consumo energético eficiente para aplicaciones conectadas

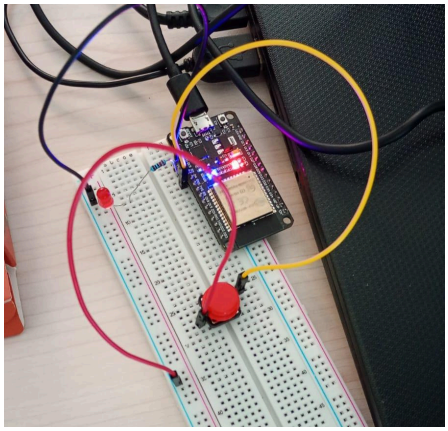
Se integró un pulsador táctil de 4 pines como dispositivo de entrada para alternar el estado del sistema manualmente.

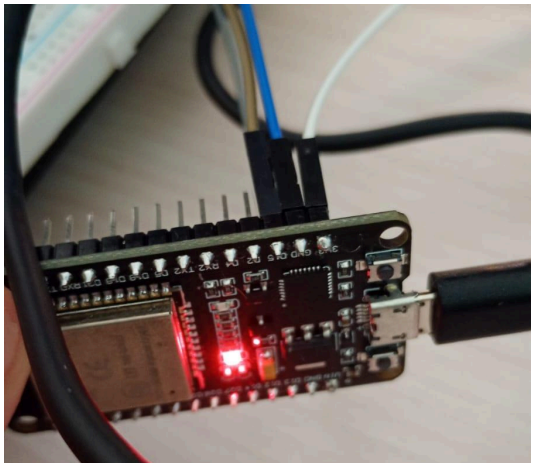
- Configuración electrónica: Como se observa en el montaje físico, el pulsador se conectó utilizando una configuración de lógica negativa. Un terminal del botón va hacia el pin GPIO del microcontrolador (mediante cable amarillo) y el terminal opuesto se conecta directamente a la línea de tierra/GND (mediante cable rojo).
- Justificación del INPUT_PULLUP: Debido a que el botón cierra el circuito hacia tierra, fue necesario configurar el pin en el código como INPUT_PULLUP. Esto activa la resistencia interna del ESP32 (de aprox. 45kΩ) manteniendo el pin en estado ALTO (3.3V) cuando el botón está suelto, y asegurando un estado BAJO (0V) limpio y sin "ruido" eléctrico al presionarlo . Esto eliminó la necesidad de colocar resistencias físicas externas en la protoboard.

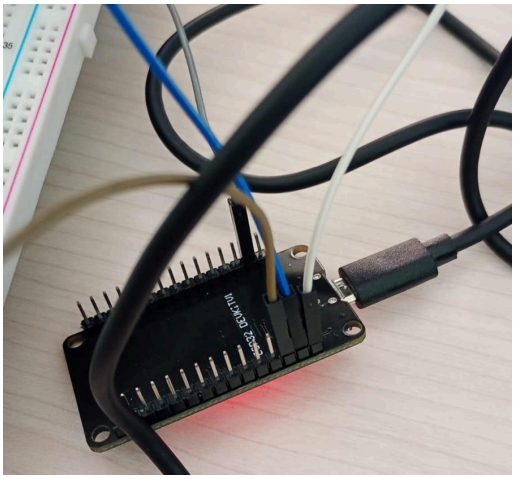
Como actuador visual para representar el estado ("Encendido/Apagado") se utilizó un diodo LED rojo.

- Conexión: El ánodo (pata positiva) se conectó al pin GPIO de salida mediante un cable azul. El cátodo se conectó a tierra (GND) pasando a través de una resistencia de protección.
- Protección del circuito: Se evidencia en el montaje el uso de una resistencia (código de colores visible) en serie con el LED. Este componente es crítico para limitar la corriente que circula desde el pin del ESP32 (3.3V), evitando que el LED se queme por exceso de amperaje .
- Control: El control se realiza mediante la función digitalWrite(), enviando un nivel lógico ALTO (HIGH) para iluminar el LED.









ADMINISTRACIÓN
Y NEGOCIOS

Inacap

4 Configuración FireBase a Aplicación

Para iniciar la integración, se creó un proyecto en la consola de Firebase con los siguientes identificadores únicos, los cuales vinculan la lógica de la nube con nuestra aplicación cliente:

- Project ID: cloudled-89946
- Nombre del Paquete (Package Name): cl.inacap.cloudled
- Plataforma: Android

La pieza clave para la conexión es el archivo google-services.json. Este archivo contiene todas las credenciales y rutas de acceso necesarias para que la API de Android se comunique con los servidores de Google sin necesidad de hardcodear (escribir manualmente) claves sensibles en el código Java/Kotlin.

Análisis del archivo implementado (google-services.json):

- Identificación del Cliente: El archivo confirma que la aplicación autorizada es cl.inacap.cloudled. Esto asegura que solo nuestra aplicación compilada pueda escribir o leer de la base de datos bajo las reglas de seguridad establecidas.
- Ruta de la Base de Datos: Se estableció automáticamente la URL de referencia para la base de datos en tiempo real:

firebase_url: https://cloudled-89946-default-rtdb.firebaseio.com

- Credenciales de API: Se integró la current_key (API Key) necesaria para validar las peticiones HTTP que salen desde el dispositivo móvil hacia la nube.

- Bucket de Almacenamiento: También se dejó configurada la ruta para almacenamiento de archivos (Storage Bucket) en cloudled-89946.firebasestorage.app, lo que permitirá escalar el proyecto si se requiere subir imágenes o archivos en el futuro.

Para que el archivo google-services.json funcionara, se realizaron los siguientes pasos en el entorno de desarrollo:

- Se colocó el archivo JSON en la carpeta raíz /app del proyecto.
- Se modificaron los archivos build.gradle (tanto a nivel de proyecto como a nivel de módulo) para incluir el plugin de Google Services (com.google.gms.google-services).
- Se añadieron las dependencias de la librería de Firebase Realtime Database.

5 Lógica del Firmware (ESP32)

Para lograr la sincronización bidireccional descrita en los objetivos, se desarrolló lógica tanto en el microcontrolador como en la aplicación móvil. A continuación se detalla el funcionamiento de cada nodo.

5.1. Firmware del Nodo IoT (ESP32)

El código cargado en el ESP32 tiene como función principal actuar como puente entre el mundo físico y la nube. Se utilizaron las librerías nativas de `WiFi.h` para la conexión a red y la librería cliente de Firebase para la gestión de datos.

El flujo de trabajo del firmware es el siguiente:

- Inicialización: Al encenderse, el dispositivo configura el pin del botón como entrada con resistencia interna (INPUT_PULLUP) y el pin del LED como salida.
- Conexión: Se establece la conexión a la red Wi-Fi y posteriormente se autentica con el proyecto cloudled-89946 utilizando el host y el auth token definidos.
- Bucle Principal (Loop):
    - Lectura Local: El sistema monitorea constantemente el estado del botón físico. Al detectar una pulsación (estado LOW), invierte el estado actual de la variable de control y envía el nuevo valor a la base de datos mediante la función de escritura (setInt o setBool).
    - Sincronización Remota: Paralelamente, el ESP32 consulta o se suscribe a cambios en la ruta /prueba_iot/estado. Si recibe un cambio de valor desde la nube (provocado por la App), actualiza inmediatamente el estado del LED físico (HIGH/LOW) para coincidir con el dato virtual.

5.2. Lógica de la Aplicación Android

La aplicación móvil actúa como la interfaz de control de usuario (HMI). Tras la configuración del archivo google-services.json detallada en la sección anterior, la lógica de comunicación se implementó en la actividad principal (MainActivity).

- Referencias a la Base de Datos: Se instanció un objeto DatabaseReference apuntando específicamente a la ruta hija prueba_iot.
- Lectura en Tiempo Real: Se implementó un listener (addValueEventListener) que permanece "escuchando" cualquier modificación en la base de datos.
- Funcionamiento: Cuando el ESP32 cambia el valor en la nube (ej. al presionar el botón físico), este método se dispara automáticamente (onDataChange), permitiendo que la App actualice el color o texto del botón en la pantalla del celular casi instantáneamente, reflejando el estado real del hardware.
- Escritura de Datos: Al interactuar con el botón en la pantalla táctil, la aplicación envía una instrucción setValue() a Firebase, sobrescribiendo el valor actual (ej. de "0" a "1"). Esto cierra el ciclo de comunicación bidireccional.

5. Pruebas y Resultados

Se realizaron pruebas de campo para validar la robustez del sistema y la velocidad de respuesta (latencia).

Prueba 1: Control desde la App Se pulsó el botón virtual en la aplicación Android.

- Resultado: El valor en Firebase cambió instantáneamente y el LED físico conectado al ESP32 se encendió en menos de 1 segundo.
- Evidencia: [Insertar foto: Celular mostrando la App activada junto al protoboard con el LED encendido]

Prueba 2: Control Físico y Sincronización Se presionó el pulsador físico en el protoboard utilizando la lógica de INPUT_PULLUP.

- Resultado: El LED cambió de estado y la interfaz de la aplicación se actualizó automáticamente sin necesidad de recargar la pantalla, validando el funcionamiento del listener en Android.

Prueba 3: Persistencia de Conexión Se desconectó la alimentación del ESP32 y se volvió a conectar.

- Resultado: Al reiniciarse, el dispositivo consultó el último estado en la nube y encendió/apagó el LED acorde a la información guardada, cumpliendo con el objetivo de recuperación de estado.

6 Conclusión

La ejecución del proyecto CloudLed ha permitido cumplir satisfactoriamente con todos los objetivos planteados, demostrando la eficacia de la arquitectura seleccionada para soluciones IoT de baja latencia.

En primer lugar, la integración entre la aplicación Android y el microcontrolador ESP32 mediante Firebase Realtime Database resultó ser robusta y eficiente. La base de datos NoSQL permite una sincronización bidireccional casi instantánea (tiempo real), donde los cambios físicos se reflejan en la interfaz de usuario y los comandos remotos actúan sobre el hardware sin retardos perceptibles.

Desde el punto de vista del hardware, la implementación de la lógica con resistencia interna INPUT_PULLUP validó ser una optimización clave. Esto no sólo simplificó el circuito eliminando resistencias externas en la protoboard, sino que garantizó estados lógicos estables (HIGH/LOW), eliminando el "ruido" eléctrico y asegurando una lectura fiable del pulsador.

Finalmente, las pruebas de campo confirmaron la resiliencia del sistema. La capacidad del ESP32 para consultar el estado en la nube al iniciarse asegura que el dispositivo siempre recupere su última configuración operativa, incluso tras cortes de energía. En resumen, el sistema CloudLed sienta una base sólida y escalable para futuros desarrollos domóticos más complejos, validando el uso de herramientas modernas como Android Studio y servicios en la nube de Google.