IoT con ESP32: Conectividad y Plataformas

El proyecto "IoT con ESP32 y Arduino" se pretende mostrar una aproximación exhaustiva al Internet de las Cosas (IoT) utilizando el microcontrolador ESP32, cubriendo tanto la infraestructura de comunicación como las herramientas en la nube para el manejo de datos de forma remota.

A continuación, se presenta un análisis de los puntos claves del proyecto, basado en la conectividad y las plataformas:

I. El Microcontrolador y sus Capacidades de Conectividad

El proyecto se centra en el **ESP32** debido a su bajo costo y versatilidad, destacando sus características superiores en comparación con el ESP8266, como su procesador de doble núcleo (hasta 240 MHz) y la inclusión de Bluetooth (una ventaja que el ESP8266 nativo no tiene).

Métodos de Conectividad Explorados:

1. **Wi-Fi y Redes:** El ESP32 es compatible con los estándares WiFi de 2.4 GHz (802.11b/g/n). El módulo puede configurarse en **Modo Estación (cliente)**, conectándose a un router doméstico, o en **Modo Punto de Acceso (AP)**, creando su propia red. También se puede operar en modo mixto o híbrido (APSTA).

2. **Gestión Dinámica de Redes:** Para evitar la tediosa necesidad de editar y regrabar el código cada vez que se cambia de red, el proyecto utilizara la biblioteca **WiFi Manager**. Esta permitira al ESP32 buscar una red preestablecida o, si falla, activar el modo AP para que el usuario ingrese manualmente las credenciales de Wi-Fi a través de una página de configuración HTML, guardando los datos en la EEPROM.

3. **Comunicación Serial y Bluetooth:** El proyecto incluira el manejo de comunicación serial (TX/RX), SPI, e **I2C**. También se cubrira el **control remoto vía Bluetooth** utilizando una aplicación diseñada en **App Inventor** para encender y apagar LEDs desde un dispositivo móvil.

4. **Protocolos IoT:** Se utilizaran diversos protocolos esenciales para IoT, como **TCP/IP, UDP, y MQTT**.

5. **Actualización Inalámbrica (OTA):** El proyecto abordara la **actualización de firmware vía Wi-Fi (OTA)** sin necesidad de cables, utilizando la biblioteca `Elegant OTA` junto con `Async Web Server`. Esto permitira actualizar el código del ESP32 de forma remota, aunque necesita autenticación (usuario/clave) en la red local para la seguridad.

Consideración Crítica del Hardware:

Un detalle importante a tener en cuenta en la conectividad es el nivel lógico. El ESP32 solo funciona a **3.3V**. Se advierte explícitamente que conectar periféricos (como LCDs, sensores o módulos) de 5V directamente al ESP32 puede dañarlo permanentemente. Para mitigar esto, se recomienda el uso de **convertidores de nivel lógico** (como el módulo TXS0108E).

II. Plataformas y Servicios IoT (La Nube)

El proyecto distingue entre el uso de un **servidor web local** (Web Server HTTP) y el uso de **plataformas IoT en la nube** (servicios remotos).

1. Plataformas IoT Genéricas

Se presentara una variedad de plataformas que ya ofrecen una estructura lista para visualizar datos:

- **Ejemplos: ThingSpeak**, **Ubidots**, **Adafruit**, **TagoIO**, **AWS IoT**, y **Azure IoT**.
- **Características:** Estas plataformas generan una API y una autenticación para que el ESP32 se conecte. Ofrecen reproyectos gráficos como botones, relojes y gráficas para la visualización.
- **Limitaciones:** Se señala que las versiones gratuitas de estos servicios suelen ser limitadas, por ejemplo, restringiendo la cantidad de datos que se pueden subir al día o eliminando la base de datos después de cierto tiempo.

2. Google Firebase (Foco Principal del Curso)

Firebase, de Google, sera la plataforma principal utilizada para el almacenamiento y sincronización de datos, siendo reconocida por ser **multiplataforma** y por facilitar el desarrollo de aplicaciones (iOS, Android, web).

- **Realtime Database:** Es el componente principal utilizado para almacenar datos en la nube. Los datos se almacenaran en formato **JSON** (JavaScript Object Notation).
- **Sincronización:** Firebase utiliza la **sincronización de datos**; cada vez que los datos cambian, los dispositivos conectados recibiran la actualización en milisegundos, ofreciendo experiencias colaborativas y envolventes.
- **Configuración y Autenticación:** La conexión del ESP32 a Firebase requierira de dos elementos claves: la **API Key** y la **URL de la base de datos**. El proyecto aborda la evolución de la autenticación:
  - Inicialmente, se usara el **modo de prueba o anónimo** para demostración.
  - Posteriormente, se implementara una autenticación más robusta por **correo electrónico y contraseña**, junto con reglas específicas que autorizaran la lectura y escritura para el usuario autenticado (UID).

3. Desarrollo de Aplicaciones y Visualización Web

El proyecto va más allá del uso de interfaces preestablecidas y enseñara a crear soluciones personalizadas:

- **App Inventor:** Se utilizara para **diseñar aplicaciones móviles** (Android/iOS) que se conectaran a Firebase para permitir el control a distancia (ejemplo: encendido/apagado de LEDs). Se aclara que App Inventor tiende a enviar datos como cadenas de caracteres (**String**) en lugar de números, lo que requiere una adaptación en el código de Arduino para convertir el *String* recibido a *Integer*.
- **Sitio Web Personalizado (Firebase Hosting):** Para crear una interfaz de monitoreo y control a nivel profesional, el proyecto utilizara **Visual Code Studio**, **Node.js**, y **Firebase Hosting**. Esto permite diseñar una página web personalizada con HTML, CSS, y JavaScript, haciendo posible la visualización de datos de sensores (como el Conversor Analógico Digital, ADC) en tiempo real, accesible mediante un link y autenticación por correo/clave.

III. Adquisición de Datos (ADC)

Un componente fundamental de la conectividad en el proyecto es el uso del Conversor Analógico Digital (ADC) del ESP32 para adquirir señales de sensores (ejemplo: potenciómetros).

- **Características:** El ESP32 incluira **dos ADCs** (ADC1 y ADC2), siendo el **ADC1** el más recomendado ya que el ADC2 suele presentar inconvenientes cuando se utiliza el modo Wi-Fi. La resolución máxima es de **12 bits**, lo que se traduce en un rango de lectura de 0 a 4095 bits, correspondiente a 0 a 3.3V.

- **Modificación de Resolución:** Es posible modificar la resolución del ADC (por ejemplo, a 10 bits como en Arduino convencional) mediante la instrucción `analogReadResolution`, aunque la recomendación es aprovechar la máxima capacidad de 12 bits.
- **Uso Práctico:** El valor leído del ADC se convierte a voltaje (de 0 a 3.3V) y luego se almacena en la base de datos de la plataforma elegida (Firebase o ThingSpeak).

En conclusión, el proyecto proporcionara una **ruta completa** para el desarrollo de proyectos IoT, iniciando con la programación básica del ESP32, pasando por la gestión inteligente de la conectividad Wi-Fi, y culminando con el uso avanzado de plataformas en la nube (especialmente Firebase) para el monitoreo y control a través de aplicaciones móviles y sitios web personalizados.