05 NodeJS: Johnny-five

Hasta ahora, hemos discutido Node.js principalmente en el contexto del desarrollo de servidores web, APIs y herramientas de línea de comandos. Sin embargo, para demostrar la amplitud de sus capacidades, es útil explorar un campo de aplicación diferente: la **robótica y el Internet de las Cosas (IoT)**. Un excelente ejemplo de cómo Node.js facilita la interacción con hardware físico es la plataforma **Johnny-Five**.

¿Qué es Johnny-Five?

Johnny-Five es una plataforma de programación de Robótica y IoT de código abierto, escrita en JavaScript, que se ejecuta sobre Node.js. Su objetivo es simplificar el proceso de experimentar y construir proyectos que involucren hardware electrónico, como microcontroladores (por ejemplo, Arduino, Raspberry Pi, Tessel 2, Intel Edison), sensores, LEDs, motores y otros componentes.

¿Cómo funciona con Node.js?

1. **Entorno de Ejecución:** Johnny-Five es, fundamentalmente, un módulo de Node.js que se instala a través de npm (`npm install johnny-five`). El código que escribes utilizando Johnny-Five es código JavaScript que se ejecuta mediante el *runtime* de Node.js en tu ordenador (o directamente en dispositivos compatibles como Raspberry Pi).
2. **Abstracción de Hardware:** Proporciona una API (Interfaz de Programación de Aplicaciones) expresiva y de alto nivel que abstrae las complejidades de la comunicación de bajo nivel con diferentes tipos de hardware. En lugar de escribir código específico para cada placa o protocolo (como Firmata, que a menudo se usa para comunicar con Arduino), puedes usar una sintaxis JavaScript consistente y legible.
3. **Modelo Orientado a Eventos:** Se alinea perfectamente con la naturaleza asíncrona y orientada a eventos de Node.js. Puedes escribir código que reaccione a eventos del hardware, como un botón que se presiona, un sensor que detecta movimiento o un potenciómetro que cambia de valor, utilizando el familiar sistema de *callbacks* o eventos de JavaScript.

¿Qué permite hacer?

Con Node.js y Johnny-Five, un desarrollador JavaScript puede:

- Controlar el encendido y apagado de LEDs, variar su brillo.
- Leer datos de sensores (temperatura, humedad, luz, distancia, movimiento).
- Controlar motores (servomotores, motores DC).
- Interactuar con pantallas LCD.
- Leer entradas de botones, interruptores, potenciómetros.
- Orquestar interacciones complejas entre múltiples componentes de hardware.

Ejemplo Conceptual (Control de un LED):

Imaginemos que queremos hacer parpadear un LED conectado a una placa Arduino, la cual está conectada a nuestro ordenador donde se ejecuta Node.js. El código Johnny-Five podría parecerse a esto (simplificado conceptualmente):

```
const five = require("johnny-five");
const board = new five.Board(); // Intenta conectar con la placa (ej. Arduino)

board.on("ready", () => {
  // Una vez la placa está lista...
  console.log("Placa lista. Iniciando parpadeo...");

  // Crea un objeto Led asociado al pin digital 13
  const led = new five.Led(13);

  // Hace parpadear el LED cada 500 milisegundos
  led.blink(500);

  // Se puede acceder al LED desde el REPL para control interactivo
  board.repl.inject({
    led: led
  });
});

board.on("error", (err) => {
  console.error("Error de conexión con la placa:", err);
});
```

En este ejemplo, se observa cómo se utiliza la sintaxis de JavaScript (`require`, objetos, métodos como `.on()`, `.blink()`, funciones de flecha) para controlar un componente físico.

> En el Arduino habría de flashearse un *firmware* específico, siendo el más común **StandardFirmata** (o una de sus variantes). Firmata es un protocolo genérico para comunicarse con microcontroladores desde software en un ordenador *host*. Una vez que el firmware Firmata está en el Arduino, actúa como un intermediario: escucha comandos enviados a través del puerto serie (USB) desde el ordenador y los traduce en acciones en sus pines (encender un LED, leer un sensor, mover un servo, etc.), y también envía datos de los sensores de vuelta al ordenador.

> Johnny-Five es compatible con una amplia variedad de placas, incluyendo las basadas en el ESP32. El principio es similar: flasheas un firmware compatible con Firmata en el ESP32. Dada la capacidad WiFi del ESP32, es muy común usar una versión como **StandardFirmataWiFi**.

> Esto permite que tu código Johnny-Five (ejecutándose en Node.js en tu ordenador o Raspberry Pi) se comunique con el ESP32 **a través de la red WiFi**, en lugar de requerir una conexión USB directa (aunque la

> comunicación por USB/Serie también suele ser posible). Esto abre muchas posibilidades para proyectos IoT inalámbricos controlados con JavaScript.

¿Por qué es relevante este ejemplo?

- **Demuestra la versatilidad de Node.js:** Muestra que Node.js no se limita a la web, sino que puede actuar como puente entre el software y el mundo físico.
- **Refuerza el poder del ecosistema npm:** Johnny-Five es un ejemplo de cómo la comunidad de Node.js extiende las capacidades de la plataforma a través de módulos.
- **Ilustra el modelo de eventos en acción:** La respuesta a eventos (`board.on("ready", ...)`), aunque simple aquí, es fundamental en aplicaciones IoT más complejas (ej. reaccionar a un sensor).
- **Accesibilidad para desarrolladores JS:** Permite a programadores con experiencia en JavaScript adentrarse en el campo del hardware sin necesidad de aprender lenguajes de bajo nivel (como C/C++) de inmediato para prototipos o proyectos sencillos.

En definitiva, Johnny-Five es una manifestación práctica de cómo Node.js sirve como un entorno flexible que permite aplicar las habilidades de programación en JavaScript a una gama diversa de problemas, incluyendo la interacción con el hardware.