



TECNOLÓGICO  
NACIONAL DE MÉXICO



TECNOLÓGICO NACIONAL DE MÉXICO

## INSTITUTO TECNOLÓGICO DE ORIZABA

### MATERIA

TECNOLOGÍAS E INTERFACES DE COMPUTADORAS

### DOCENTE

NORMA RODRIGUEZ RODRIGUEZ

### HORARIO

13:00 – 14:00

### INTEGRANTES DEL EQUIPO

GARCIA DAMIAN BEATRIZ ANDREA

GARCIA HERNANDEZ RICARDO

GONZALEZ FRANCO XIMENA

GONZALEZ JIMENEZ ALEXIS

### UNIDAD

3

### NOMBRE DE LA ACTIVIDAD

ACTIVIDAD 1. INVESTIGACIÓN

### FECHA DE ENTREGA

MARTES, 11 DE NOVIEMBRE DEL 2025





## INTRODUCCIÓN

La comunicación entre una computadora y dispositivos externos constituye un elemento fundamental para el intercambio de información y el control de hardware. Esta interacción se realiza comúnmente mediante puertos estándar como el puerto serial (COM o USB virtual), que permiten establecer una transferencia de datos confiable entre sistemas.

Para lograr dicha comunicación, los lenguajes de programación actuales ofrecen diversas herramientas y bibliotecas que facilitan el manejo de puertos, eliminando la necesidad de manipular directamente las configuraciones de bajo nivel del hardware.

El objetivo de esta actividad es investigar y analizar tres herramientas y bibliotecas modernas empleadas para la gestión de puertos de comunicación: Node-SerialPort, Johnny-Five y Boost.Aasio. Cada una pertenece a un entorno de programación diferente (JavaScript/Node.js y C++), lo que permite observar distintos enfoques para resolver un mismo problema técnico. A lo largo del desarrollo, se describen las características, ventajas, limitaciones y casos de uso de cada una, resaltando su importancia en el desarrollo de aplicaciones con transferencia de datos, sistemas de monitoreo, automatización y proyectos de Internet de las Cosas (IoT).

Estas herramientas representan la evolución de las interfaces de comunicación entre software y hardware, mostrando cómo la programación moderna ha simplificado el acceso a los puertos físicos mediante bibliotecas eficientes, multiplataforma y con soporte asíncrono, permitiendo a los desarrolladores construir soluciones cada vez más flexibles y potentes.



### ACTIVIDAD 1:

Investigar 3 diferentes herramientas y bibliotecas utilizadas para gestionar puertos como PySerial en Python, Java Communications API, etc. y presentar sus ventajas, limitaciones. y casos de uso.

#### 1) Node-SerialPort (JavaScript / Node.js)

**Node-SerialPort**, conocido también como **SerialPort**, es una **biblioteca de JavaScript** que se utiliza dentro del entorno de ejecución **Node.js** para permitir la **comunicación a través de puertos seriales**.

Esta biblioteca facilita la conexión entre computadoras y dispositivos externos (como microcontroladores, sensores o tarjetas Arduino) mediante el envío y recepción de datos seriales.



Node-SerialPort forma parte del ecosistema oficial de Node.js y se distribuye como un paquete de **npm (Node Package Manager)**. Proporciona una API moderna basada en eventos y streams, lo que permite un manejo eficiente y asíncrono de la información. Además, cuenta con parsers integrados que permiten separar los datos recibidos por líneas, bytes o delimitadores personalizados.

### HERRAMIENTA Y BIBLIOTECA

- **Herramienta:** Node.js (entorno que permite ejecutar JavaScript fuera del navegador).
- **Biblioteca:** Node-SerialPort (módulo que se instala con npm install serialport).

### VENTAJAS

- **Multiplataforma:** funciona en Windows, Linux y macOS sin necesidad de configuraciones complicadas.
- **Instalación sencilla:** se instala fácilmente con npm.
- **API basada en eventos:** permite trabajar con eventos como “open”, “data”, “error” y “close”.
- **Integración web:** puede usarse junto con otras bibliotecas o frameworks de Node.js para enviar datos a servidores o interfaces web en tiempo real.
- **Soporta parsers:** permite manejar flujos de datos complejos con parsers personalizados (por línea, byte, delimitador, etc.).

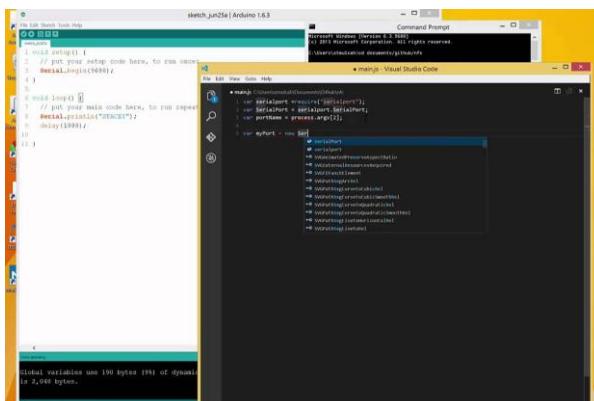


## LIMITACIONES

- **Dependencia de bindings nativos:** en algunos sistemas operativos, puede requerir permisos de administrador o librerías adicionales al instalar.
- **No recomendado para tareas de tiempo real estricto:** aunque eficiente, Node.js no garantiza la precisión temporal que requieren algunos sistemas industriales críticos.
- **Curva de aprendizaje en asincronía:** requiere entender el modelo de programación asíncrono de Node.js.

## CASOS DE USO

- **Comunicación entre una computadora y una placa Arduino**  
Node-SerialPort permite enviar instrucciones desde una aplicación Node.js a una placa Arduino a través del puerto USB. Por ejemplo, se puede programar una interfaz web que controle luces LED o motores en tiempo real desde el navegador.  
El servidor Node.js lee los datos del puerto serial y los envía mediante **WebSockets** a una página web que muestra los valores o estados de los dispositivos conectados.
- **Monitoreo de sensores y registro de datos**  
Es muy útil para proyectos de **monitoreo ambiental o industrial**, donde un sensor envía información a través de un puerto serie. Node.js puede recibir esos datos continuamente (por ejemplo, temperatura, humedad o nivel de gas) y almacenarlos en una base de datos o en la nube.
- **Interfaz de usuario para dispositivos embebidos**  
Con Node-SerialPort se pueden crear programas que muestren información en tiempo real desde un microcontrolador (por ejemplo, datos de un GPS o un módulo GSM). El usuario puede enviar comandos desde la PC y recibir respuestas inmediatas del dispositivo.
- **Automatización de pruebas de hardware**  
En entornos de desarrollo electrónico, se usa para **automatizar pruebas de dispositivos** conectados por puerto serial, permitiendo a los ingenieros validar firmware, leer logs de depuración o enviar comandos de diagnóstico automáticamente.





## 2) Johnny-Five (JavaScript / Node.js)

**Johnny-Five** es una **biblioteca o framework de JavaScript** desarrollada también para el entorno **Node.js**, orientada al control de hardware mediante una API de alto nivel. Su objetivo principal es permitir a los desarrolladores programar dispositivos electrónicos como **placas Arduino, Raspberry Pi, Tessel, BeagleBone, Particle Photon**, entre otros, usando JavaScript.

Johnny-Five se comunica con los microcontroladores principalmente a través del protocolo **Firmata**, que se ejecuta en el dispositivo. Internamente utiliza la biblioteca **Node-SerialPort** para la comunicación física con los puertos seriales, pero ofrece una interfaz mucho más sencilla y orientada a objetos.



### HERRAMIENTA Y BIBLIOTECA

- **Herramienta:** Node.js.
- **Biblioteca:** Johnny-Five (instalable desde npm con npm install johnny-five).

### VENTAJAS

- **Fácil de usar:** permite controlar hardware con unas pocas líneas de código.
- **Basado en JavaScript:** accesible para quienes ya dominan la programación web.
- **Multiplataforma:** funciona en múltiples sistemas operativos.
- **Amplio soporte de hardware:** compatible con decenas de placas y sensores.
- **Enorme comunidad:** existe mucha documentación, ejemplos y soporte en foros.

### LIMITACIONES

- **Depende de Node-SerialPort:** no puede funcionar sin la comunicación física serial.
- **No es ideal para proyectos industriales complejos:** se orienta más a educación, prototipado y robótica ligera.



- **Requiere firmware Firmata en la placa:** sin él, no puede comunicarse correctamente.
- **Latencia moderada:** no es ideal para tareas que requieren control en tiempo real milisegundo a milisegundo.

## CASOS DE USO

- **Robótica educativa y prototipado rápido**  
Johnny-Five es ampliamente usado en **proyectos de enseñanza y aprendizaje de robótica**, ya que permite controlar motores, servos, LEDs o sensores con unas pocas líneas de código JavaScript. Por ejemplo, los estudiantes pueden construir un robot que siga líneas o esquive obstáculos, y controlarlo desde una interfaz web sin tener que escribir código en C o C++.
- **Internet de las Cosas (IoT)**  
Se puede usar Johnny-Five junto con plataformas como **MQTT** o **Firebase** para enviar datos de sensores a la nube y visualizarlos desde cualquier lugar. Por ejemplo, una estación meteorológica conectada a un Arduino puede enviar información de temperatura y humedad a un servidor Node.js que la procesa y la muestra en una aplicación web.
- **Control remoto de sistemas físicos**  
Es ideal para crear **sistemas de automatización doméstica** (encender luces, abrir puertas, medir niveles de agua, etc.). Gracias a su integración con frameworks de Node.js, se pueden crear paneles de control interactivos que se comuniquen directamente con el hardware.
- **Prototipos experimentales y artísticos**  
Johnny-Five también se utiliza en proyectos de arte interactivo o instalaciones tecnológicas donde se requiere conectar hardware (luces, servos, sensores de movimiento) a un software creativo que responde a estímulos físicos o de sonido.

The screenshot shows the Cloud9 IDE interface. On the left, the file tree displays a directory structure for a project named 'Cloud9'. Inside 'Cloud9', there are several subfolders like 'cloud9', 'c9', '\_includes', 'autorun', 'beaglebone101', 'examples', 'morphocode', and 'node\_modules'. Under 'node\_modules', the file 'led-blink.js' is selected. The main workspace shows the code for 'led-blink.js':

```
1 var five = require("johnny-five"),
2 BeagleBone = require("beaglebone-io");
3
4 var board = new five.Board({
5   io: new BeagleBoneIO
6 });
7
8 board.on("ready", function () {
9   // pin 5 is mapped to the P9_14 pin of the beagleboard
10  var led = new five.Led("P9_14");
11
12  led.blink(1000);
13
14  // this allows access to the led while the program is running
15  this.repl.inject({ led: led });
16 });
17
```

The terminal at the bottom of the IDE shows the command 'morphocode led-blink' being run, followed by the output:

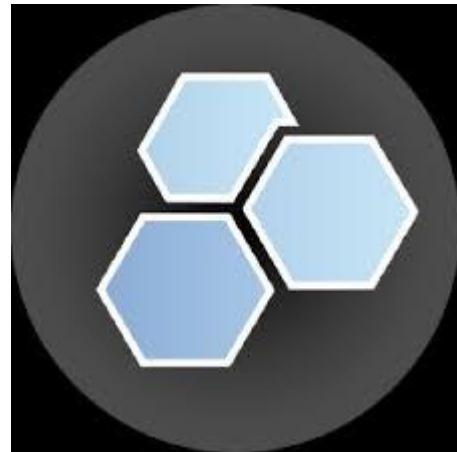
```
1425242997273 Device(s) BeagleBone-10
1425242997333 Connected BeagleBone-10
1425242997410 Repl Initialized
```



### 3) Boost.Asio (C++)

**Boost.Asio** es una **biblioteca de C++** incluida en el conjunto de librerías **Boost**, que ofrece herramientas para **programación de entrada/salida (I/O) asíncrona**.

Permite manejar **comunicaciones de red y puertos seriales** de manera eficiente y portable entre sistemas operativos como Windows, Linux y macOS. Dentro de Boost.Asio se encuentra la clase `boost::asio::serial_port`, que facilita el control directo de los puertos serie mediante operaciones síncronas y asíncronas.



Esta biblioteca se utiliza ampliamente en aplicaciones profesionales que requieren **altas velocidades de comunicación y bajo uso de recursos**, como software industrial, sistemas embebidos, telecomunicaciones y simuladores.

#### HERRAMIENTA Y BIBLIOTECA

- **Herramienta:** C++ (lenguaje y compilador, como GCC o MSVC).
- **Biblioteca:** Boost.Asio (parte del proyecto Boost).

#### VENTAJAS

- **Rendimiento muy alto:** escrita en C++, ofrece bajo consumo y gran velocidad.
- **Soporte de asincronía:** permite manejar múltiples puertos o conexiones simultáneamente sin bloquear el programa.
- **Multiplataforma:** misma API en Windows, Linux y macOS.
- **Gran estabilidad:** usada en proyectos profesionales y de código abierto.
- **Integración con otros componentes Boost:** interoperable con otras bibliotecas del mismo conjunto.

#### LIMITACIONES

- **Mayor complejidad:** requiere conocimientos intermedios o avanzados de C++.
- **Configuración inicial más técnica:** es necesario compilar el proyecto y enlazar correctamente las librerías Boost.



- **No tan rápida de implementar:** comparada con soluciones de alto nivel como PySerial o Node-SerialPort.

## CASOS DE USO

- **Aplicaciones industriales de control y supervisión**

Boost.Asio se utiliza en **sistemas de control industrial (SCADA)**, donde múltiples dispositivos (como controladores lógicos programables, sensores o actuadores) deben comunicarse mediante puertos serie RS-232 o RS-485 con alta confiabilidad y bajo retardo.

Por su rendimiento, se emplea en plantas de manufactura, sistemas energéticos y automatización de procesos.

- **Sistemas embebidos y comunicación entre microcontroladores y PC**

En entornos de desarrollo de hardware, Boost.Asio permite que una aplicación C++ se comunique con un microcontrolador (por ejemplo, un STM32 o un ESP32) para enviar y recibir datos de control, configuraciones o resultados de sensores en tiempo real.

- **Simuladores o instrumentos científicos**

Muchos simuladores industriales o científicos utilizan Boost.Asio para **intercambiar datos de precisión** con dispositivos externos. Por ejemplo, se puede comunicar un software de simulación de vuelo con un panel de instrumentos físico conectado por puerto serie.

- **Aplicaciones de red combinadas con comunicación serial**

Dado que Boost.Asio maneja tanto puertos serie como sockets TCP/IP, es posible crear programas híbridos que reciban datos de sensores por puerto serial y los reenvíen a una red o servidor remoto. Esto es útil en sistemas de telemetría o monitoreo distribuido.

```
#include <asio>
#include <iostream>

int main()
{
    boost::asio::io_service ioservice;
    boost::asio::steady_timer timer1(ioservice, std::chrono::seconds(2));
    timer1.async_wait([](const boost::system::error_code &ec) {
        std::cout << "2 Sekunden sind um!" << std::endl;
    });
    ioservice.run();
}
```



## CONCLUSIÓN

El estudio de las herramientas y bibliotecas Node-SerialPort, Johnny-Five y Boost.Asio demuestra que, aunque todas persiguen el mismo propósito —facilitar la comunicación entre computadoras y dispositivos externos—, cada una lo aborda desde distintos niveles de abstracción y complejidad. Mientras Node-SerialPort se centra en el control directo del puerto serial, Johnny-Five simplifica el trabajo mediante una interfaz de alto nivel que permite manipular componentes electrónicos con facilidad. Por su parte, Boost.Asio ofrece un enfoque de alto rendimiento y precisión, ideal para aplicaciones profesionales o industriales que requieren comunicación estable y eficiente.

Estas tecnologías evidencian el papel esencial que desempeñan las bibliotecas modernas en el desarrollo de sistemas interactivos y conectados. Gracias a ellas, es posible integrar hardware y software de manera práctica, estableciendo una base sólida para proyectos de automatización, monitoreo, control remoto e Internet de las Cosas (IoT).

En conclusión, conocer y dominar este tipo de herramientas amplía las capacidades del programador y facilita la creación de soluciones tecnológicas innovadoras basadas en comunicación por puertos estándar.



## REFERENCIAS BIBLIOGRÁFICAS

- About SerialPort / Node SerialPort.* (2024, November 8). <https://serialport.io/docs/Boost.Aasio>. (n.d.). [https://www.boost.org/doc/libs/latest/doc/html/boost\\_asio.html](https://www.boost.org/doc/libs/latest/doc/html/boost_asio.html)
- Extio Technology. (2023, July 30). *Boost.Aasio: A Powerful Networking Library for C++*. Medium. <https://medium.com/@extio/boost-asio-a-powerful-networking-library-for-c-e86367c2568d>
- Johnny-Five: the JavaScript Robotics & IoT platform.* (n.d.). Johnny-Five. <https://johnny-five.io/>
- Lab: Serial Communication with Node.js – ITP Physical Computing.* (n.d.). <https://itp.nyu.edu/physcomp/labs/labs-serial-communication/lab-serial-communication-with-node-js/>
- Mehreentahir. (n.d.). *Socket Programming in C++ using boost.asio: TCP Server and Client - CodeProject*. <https://www.codeproject.com/articles/Socket-Programming-in-Cplusplus-using-boost-asio-T#comments-section>
- Nodejs, Arduino, Chart.js y SerialPort / Fazt Web.* (n.d.). faztweb.com. <https://fazt.dev/contenido/node-arduino-serialport>
- Serialport. (n.d.). *GitHub - serialport/node-serialport: Access serial ports with JavaScript. Linux, OSX and Windows. Welcome your robotic JavaScript overlords. Better yet, program them!* GitHub. <https://github.com/serialport/node-serialport>
- Tahir, M. (2018, October 28). *Socket programming in C++ using Boost.Aasio - TCP server and client.* <https://www.c-sharpcorner.com/article/socket-programming-in-cpp-using-boost-asio-tcp-server-and-client/>