# A Javascript Web Framework for Rapid Development of Applications in IoT Systems for eHealth (Un Marco Web Javascript para el Desarrollo Rápido de Aplicaciones en Sistemas IoT para eHealth)

**Abstracto:**

Bluetooth Low Energy (BLE) es actualmente la solución de red inalámbrica dominante para eHealth y deportes. Sin embargo, la mayoría de los sensores BLE requieren aplicaciones dedicadas con capacidades de desarrollo limitadas. Este documento presenta un método para el desarrollo rápido de aplicaciones en sistemas BLE IoT distribuidos para eHealth y deportes. El método se implementa como un marco web JavaScript basado en el lienzo HTML5, WebSocket y API Web Bluetooth. Este documento demuestra cómo se puede aplicar el marco para desarrollar una aplicación para monitorear la actividad física y la frecuencia cardíaca. El marco permite a los operadores de software y servicios crear, ajustar e implementar iterativamente algoritmos de filtro en sistemas BLE IoT distribuidos, sin reiniciar nodos o reiniciar programas mediante la actualización dinámica de software.

**Publicado en:**[2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)](https://ieeexplore.ieee.org/xpl/conhome/8502682/proceeding)

**Fecha de la conferencia:** 17-20 de septiembre de 2018

**Fecha añadida a IEEE *Xplore* :** 12 de noviembre de 2018

**Información del ISBN:**

**Número de acceso de INSPEC:** 18249151

**DOI:**[10.1109 / HealthCom.2018.8531124](https://doi.org/10.1109/HealthCom.2018.8531124)

**Editorial:**IEEE

**Lugar de la conferencia:** Ostrava, República Checa

**SECCIÓN I.**

## **Introducción y motivación**

Según Gartner, 8.400 millones de dispositivos se conectaron a Internet en 2017 1 . El crecimiento de IoT es una de las fuerzas impulsoras detrás de este mayor número de dispositivos conectados. Sin embargo, actualmente el control de IoT, como OneM2M [1] , activa o desactiva principalmente los sensores y, a veces, establece umbrales. Estas soluciones limitan los cambios en tiempo real y el ajuste solo a comandos básicos. Para cambios más complejos del procesamiento local en los sensores, el enfoque suele ser realizar actualizaciones de firmware por aire (OTA). El desarrollo de sensores con OTA es engorroso, requiere reinicio y existen problemas de seguridad con el envío de archivos binarios como se muestra en [2] . Además, algunos procesamientos son más adecuados para ser realizados por una puerta de enlace y no en sensores o servidores de fondo.

Hemos identificado dos problemas que deben abordarse: primero, un método general para el desarrollo rápido de aplicaciones para aplicar cambios incrementales en la funcionalidad, y segundo, un método que utiliza actualizaciones dinámicas de software para distribuir algoritmos OTA en tiempo real. Un desarrollador u operador de aplicaciones puede desarrollar de manera iterativa una aplicación, por ejemplo, configurar y distribuir algoritmos en un sistema IoT distribuido. Actualmente, los navegadores web dominantes han implementado HTML5, que incluye nuevas capacidades para visualizar, procesar y transmitir datos. La hipótesis en este documento es que HTML5 y JavaScript se pueden aplicar para desarrollar y distribuir algoritmos en un sistema IoT Bluetooth Low Energy (BLE).

El bosquejo de este documento es el siguiente. Los trabajos relacionados sobre Bluetooth, HTML5, JavaScript y otras soluciones dinámicas de actualización de software se encuentran en la Sección II . El marco, un caso de uso y los requisitos funcionales se describen en la Sección III . En la Sección IV se presentan ejemplos de desarrollo rápido y distribución de algoritmos . Los resultados se discuten en la Sección V y se resumen en la Sección VI .

**SECCION II.**

## **Trabajo relacionado**

### A. Bluetooth de baja energía

Bluetooth Low Energy (BLE) es una solución de red inalámbrica personal, dirigida principalmente a conexiones de corta duración, especificadas por Bluetooth SIG (Grupo de interés especial). Según ABI Research, habrá 16 mil millones de dispositivos Bluetooth para 2021 2 . Las posibilidades y desafíos para BLE en un entorno IoT se describen bien en [3] . Un problema importante es el acceso restringido al nivel inferior de las pilas BLE, lo que limita el uso de la funcionalidad BLE incorporada, por ejemplo, sincronización y gestión de red para desarrolladores de aplicaciones. La capa de aplicación se ha estandarizado como perfiles GATT, muchos basados ​​en el Perfil de dispositivo de salud Bluetooth (HDP) que implementa el estándar de dispositivo de salud personal IEEE 11073-20601 [4]. Estos perfiles GATT se han implementado con éxito mediante la aplicación de las normas ISO desde sensores a puertas de enlace [5] . Los métodos para monitorear y controlar el tráfico en redes BLE en la capa de aplicación también se han implementado con éxito [6]. Actualmente, los sensores BLE se usan con frecuencia en deportes para medir el pulso, la posición (GPS), la actividad, la cadencia y la fuerza. En la automatización del hogar, los sensores BLE se utilizan para controlar y controlar la luz, la temperatura, la humedad, el nivel de ruido, la calidad del aire y en la vigilancia. Además, en salud, los sensores BLE miden, entre otras cosas, el peso, la temperatura corporal, la presión arterial, la calidad del sueño, la actividad y la luz ultravioleta. Sin embargo, un problema con todos estos sensores BLE es que tienen aplicaciones dedicadas. Aunque hay aproximadamente 60 perfiles GATT adoptados públicamente, todavía existen muchos más perfiles patentados diseñados por los fabricantes de hardware de diferentes sensores BLE. Actualmente no existe una forma genérica de recuperar datos de todos estos sensores. La Web física es una solución para interactuar con objetos físicos mediante Bluetooth. Un objeto incluye un sensor Bluetooth que utiliza el protocolo de baliza Bluetooth Eddystone para transmitir un Localizador Uniforme de Recursos (URL). Esta URL apunta a una página web con la información necesaria sobre el objeto físico, como se describe en[7] . También se ha sugerido conectar balizas Bluetooth desde una página HTML [8] . Las balizas Bluetooth no resuelven el problema de que cada sensor requiera una aplicación dedicada. Sin embargo, si un navegador web pudiera conectarse directamente a un perfil GATT, entonces la URL de una baliza Bluetooth podría apuntar a una página HTML que puede conectarse al sensor BLE específico. Por lo tanto, no habría necesidad de descargar una nueva aplicación.

### B. HTML5 y JavaScript

JavaScript se creó originalmente para verificar formularios HTML antes de enviarlos a los servidores. Sin embargo, ya en 2010, Open Mobile Alliance definió una API invocable por JavaScript para la entrega dinámica de contenido que podría acceder a las API en el lado del cliente [9] . Además, existen soluciones de JavaScript para la distribución y desarrollo de algoritmos para entornos pesados ​​computables [10]. Actualmente, JavaScript es una parte esencial de Internet, y hay muchas bibliotecas, complementos y API disponibles. Las actualizaciones dinámicas de software de las aplicaciones web se realizan de forma rutinaria mediante la transmisión de código JavaScript. HTML5 es la última versión de HTML. Incluye nuevas funciones como HTML5 Canvas y WebSocket. El lienzo HTML5 se puede usar para dibujar gráficos en una página web, sin la necesidad de un complemento de terceros como Flash. WebSocket permite una comunicación bidireccional full duplex con un servidor TCP. Otra característica es la Web Bluetooth API 3 , todavía en borrador a partir de abril de 2018, pero compatible con el 59% de los navegadores actuales 4. La API web Bluetooth permite la conexión a sensores BLE directamente desde una página web HTML5. Combinando estas tres características (HTML5 Canvas, WebSocket y Web Bluetooth), es posible procesar, enviar y visualizar datos desde un sensor BLE conectado, directamente desde un navegador HTML5 compatible. Dado que la API web de Bluetooth está disponible para los servidores node.js, JavaScript en un sensor BLE también funciona en las puertas de enlace que ejecutan node.js. Tenson 2, Raspberry Pi y Edison de Intel son ejemplos de nodos de sensores que usan JavaScript y JavaScript también se pueden instalar con una huella mínima, por ejemplo, Duktape puede ejecutarse en plataformas con flash de 192 kB y 64 kB de RAM. Estos sensores o Duktape permitirían la distribución del algoritmo JavaScript a los sensores.

### C. Otras soluciones para la actualización dinámica de software

Este documento se centra en los sensores BLE en un entorno IoT, siendo la tecnología de sensores inalámbricos dominante. Serían posibles otras alternativas como Wi-Fi como se ve en [11] . Sin embargo, actualmente hay un número limitado de sensores con capacidad de Wi-Fi. Este es también el caso de los sensores ZigBee. La actualización dinámica de software para sistemas IoT está bien investigada. Un ejemplo es Contiki [12] , que es un sistema operativo para sensores IoT que proporciona la capacidad de actualizar dinámicamente módulos ejecutables durante el tiempo de ejecución. El lenguaje de programación Erlang [13] es otra solución para el desarrollo y la distribución de algoritmos en un entorno IoT. Erlang ofrece la posibilidad de actualizar de forma dinámica el código de manera tolerante a fallas durante el tiempo de ejecución. Dynamic TinyOS[14] es una extensión del conocido sistema operativo basado en módulos para hardware integrado que permite cambios de software sin interrumpir la operación del nodo sensor. Contiki, Erlang y TinyOS son soluciones atractivas para la actualización dinámica de software. Sin embargo, actualmente, ninguno de los sistemas operativos mencionados anteriormente admite los perfiles Bluetooth GATT. Por lo tanto, este documento se centra en una solución de JavaScript.

La comunicación de red para IoT también está bien explorada. El Protocolo de aplicación restringida (CoAP) y el Transporte de telemetría de Message Queue Server (MQTT) son los dos protocolos de transporte dominantes. MQTT es un protocolo de mensaje de suscripción público que requiere un intermediario de mensajes. La interoperación entre los sensores MQTT y BLE GATT es compatible con los puentes MQTT 5 . CoAP es un protocolo de aplicación RESTful que transmite datos a través de UDP. CoAP es adecuado para sensores con capacidad limitada. Para los sensores BLE GATT, se requiere una puerta de enlace para conectar los sensores a una red IP. Es esencial admitir CoAP y MQTT para los sistemas de IoT implementados.

**SECCION III.**

## **Método y marco**

El método utilizado para verificar la hipótesis fue construir un marco y evaluar la viabilidad del uso de HTML5 en combinación con JavaScript para el desarrollo y distribución de algoritmos de filtro. Al crear un marco que aproveche las nuevas características introducidas en HTML5, la hipótesis también podría probarse con los requisitos funcionales. Estos se identificaron a partir de casos de uso de Ambient Assisted Living (AAL) relacionados con eHealth.

### A. El marco

La descripción del marco se divide en tres partes: una descripción general de los componentes, funciones y flujo de trabajo para el desarrollo rápido de algoritmos como ejemplo de aplicación, y finalmente una explicación de las características de seguridad.

#### 1) Componentes del marco

Our framework (Fig. 1) consists of three components, an HTML5 gateway, and a node.js server and an HTML5 client front-end (below 800 total lines of code) 6. The node.js server acts as a web server. It also handles WebSocket connections to the HTML5 gateway and the client front-end.

Fig. 1. - Configuración del sistema del banco de pruebas que consiste en sensores BLE, una puerta de enlace que conecta los sensores BLE, un servidor node.js y un cliente front-end.


**Fig. 1.** System setup of the testbed consisting of BLE sensors, a gateway connecting the BLE sensors, a node.js server and a client front-end.

Fig. 2. - Un sensor BLE conectado a una puerta de enlace HTML5 a la izquierda (que se ejecuta en Chrome) y un cliente front-end a la derecha (que se ejecuta en Safari).

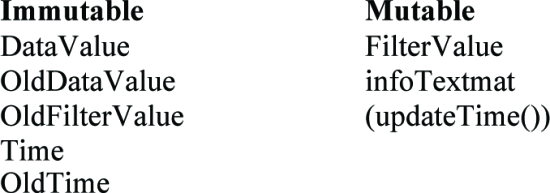

**Fig. 2.** A BLE sensor connected to an HTML5 gateway on the left (running in Chrome) and a client front-end on the right (running in Safari).

The HTML5 gateway connects and retrieves data from one or more BLE sensors utilising the Bluetooth Web API. Data from sensors are then processed locally on the gateway, using JavaScript and visualised continuously with HTML5 canvas. The testbed hardware consists of a MacBook (early 2015) running the entire framework, micro:bit 7 sensors and a pulse oximeter 8. The left part of Fig. 2 shows the gateway running on a Chrome web browser. Before the sensor is connected, the user receives a pop-up request to allow communication with the BLE sensors. In Fig. 2, a micro:bit sensor samples and transmits accelerometer data. The black line in the top left part of Fig. 2 shows raw data from an accelerometer sensor. The red line shows the raw data with a smoothing EWMA (Exponential Weighted Moving Average) filter. The programming code for the EWMA filter algorithm is displayed in the text field at the bottom left of Fig. 2.

#### 2) Function and Workflow

Para permitir un desarrollo rápido en el marco, es posible escribir o cambiar el algoritmo de filtro en el campo de texto. El algoritmo de filtro se guarda como una variable que puede cambiar sin reiniciar el programa. Para cada paquete de un sensor BLE, se ejecuta una función que incluye el algoritmo de filtro. Por razones de seguridad, un pequeño conjunto de selectores limita la codificación de algoritmos de filtro. Estos selectores se identificaron mediante el análisis del filtro digital convencional para la transmisión de datos. Estos selectores constan de siete variables ( Tabla I) Las variables inmutables son de solo lectura. Las variables que son mutables, es decir, posibles de modificar en un algoritmo de filtro, son <monospace> FilterValue </monospace> y <monospace> infoText </monospace>. <monospace> DataValue </monospace> contiene la última muestra recibida del sensor BLE. El tiempo se actualiza con la función <monospace> updateTime () </monospace>. Los selectores con el prefijo <monospace> Antiguo </monospace> contienen los valores de tiempo, filtro y muestra anteriores. Con estos selectores, es factible construir un número extendido de algoritmos de filtro. Esta limitación de un pequeño conjunto de selectores solo afecta la capacidad de crear nuevas variables o cambiar variables en la página que no sean los selectores.

**TABLA I.** Selector

[[](https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/8502682/8531078/8531124/jonas.t1-p6-jonas-large.gif)](https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/8502682/8531078/8531124/jonas.t1-p6-jonas-large.gif)

Los datos del sensor se transmiten a través de un WebSocket desde la puerta de enlace al servidor node.js. El servidor luego reenvía todos los datos al front-end del cliente HTML5. Inicialmente, los datos de la puerta de enlace son un flujo de muestras del sensor BLE conectado. La parte derecha de la Fig. 2muestra un front-end, que se ejecuta en un navegador Safari, que recibe datos de un servidor node.js. El algoritmo de filtro se ejecuta localmente en el front-end cuando se presiona el botón <monospace> Filtro de programa </monospace>. El sensor también se puede controlar de forma remota activando el botón <monospace> Enviar </monospace>, que transmite el algoritmo de filtro a la puerta de enlace. En consecuencia, la puerta de enlace ejecutará el algoritmo de filtro y comenzará a transmitir datos filtrados. El botón <monospace> ToggleRaw </monospace> inicia o detiene la puerta de enlace que envía muestras desde el sensor BLE. El entorno de desarrollo para el marco también admite la grabación de muestras (botón <monospace> Grabar </monospace>) que se pueden guardar en el archivo (botón <monospace> Guardar </monospace>). Los datos también se pueden cargar desde un archivo (<monospace> Elija el botón </monospace>) para evaluar diferentes algoritmos de filtro sin conexión. Los datos cargados se repetirán indefinidamente, lo que facilita la comparación de diferentes algoritmos de filtro. Cada transmisión de datos entre el front-end, el servidor y la puerta de enlace incluye un byte de identificación. Este byte identifica una solicitud para detener o comenzar a enviar datos sin procesar, actualizar algoritmos de filtro (el código de ejemplo está enFig. 3 ), y así sucesivamente. Actualmente, hay cuatro bytes de identificación diferentes ( Tabla II ). Sin embargo, esto se puede ampliar para cubrir más opciones, por ejemplo, actualizar el algoritmo de filtro en el nodo del sensor.

**TABLA II.**Bytes de identificación

#### Tabla II.

#### 3) seguridad

El marco tiene dos mecanismos de seguridad. La primera opción es usar HTTPS y WSS para asegurar la comunicación con TLS. En segundo lugar, el marco protege contra el código JavaScript malicioso o mal escrito mediante la ejecución de sandboxing de algoritmos de filtro. Esto se puede hacer de diferentes maneras [15] . Implementado en el marco es una solución general. La ejecución de algoritmos de filtro está envuelta dentro de una función que garantiza que solo el algoritmo de filtro afecte a los selectores.

[[Fig. 3. - 
Examples of filters using selectors.
](https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/8502682/8531078/8531124/jonas3-p6-jonas-large.gif)](https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/8502682/8531078/8531124/jonas3-p6-jonas-large.gif)

**Fig. 3.** Ejemplos de filtros que usan selectores.

### B. Desafíos del caso de uso de AAL en eHealth

Los requisitos funcionales en la Sección III.C se derivaron de los siete casos de uso representativos encontrados en los Casos de Uso y Perfiles de Integración AAL (Ambient Assisted Living) [16]. Estos casos de uso se pueden resumir como una aplicación de eHealth en la que una mujer mayor tiene una unidad móvil para controlar su actividad física y frecuencia cardíaca. Los datos de la unidad móvil se transmiten al proveedor de atención médica local para proporcionar comentarios sobre el progreso y sugerir actividad física adecuada. Estas recomendaciones la motivan y la ayudan a vivir una vida saludable. Además, la unidad móvil se conecta a sensores externos en el hogar para rastrear el comportamiento, por ejemplo, para detectar si la persona que mira televisión ha olvidado el agua hirviendo en la estufa. Con un dispensador de medicamentos conectado a la unidad móvil, es posible monitorear y recordar la ingesta de medicamentos. El sistema también puede detectar emergencias. Para estos casos de uso relacionados con la salud electrónica, Un desarrollador de aplicaciones es responsable de configurar todo el sistema desde los sensores externos a través de la unidad móvil hasta el proveedor de atención médica local. Un desafío para el desarrollador de aplicaciones es la gran diversidad de sensores y unidades móviles disponibles. El acelerómetro triaxial en un micro: bit y un oxímetro de pulso conectado midieron la actividad física y cardíaca en nuestro banco de pruebas. Sin embargo, ese es solo uno de varios tipos diferentes de equipos de hardware para rastrear la actividad física y la frecuencia cardíaca de una persona. Para un desarrollador de aplicaciones, se necesitará una configuración manual para conectar nuevo hardware. En consecuencia, una solución debe permitir la adaptación a la diversidad de hardware, la configuración y el ajuste sin la necesidad de una reprogramación y reinicio extensivos del dispositivo sensor.

### C. Requisitos funcionales

Los médicos, atletas y otros usuarios finales no están interesados ​​en datos sin procesar de un sistema de monitoreo. En cambio, los usuarios quieren datos procesados, por ejemplo, actividad física; ritmo cardiaco; cadencia; número de pasos o cuánto tiempo ha estado sentado, parado o caminando un paciente durante el día. Por lo tanto, existe la necesidad de una traducción basada en algoritmos de datos muestreados a procesados. Se requiere que los desarrolladores de aplicaciones creen y sintonicen un algoritmo adecuado para obtener información relevante al introducir nuevos sensores. Se requiere el desarrollo directamente en el sensor o desde la puerta de enlace para la configuración inicial y el desarrollo iterativo con el usuario, por ejemplo, atletas de élite. Para la configuración remota o el ajuste fino, es esencial poder continuar el desarrollo desde otras partes del sistema, por ejemplo,

Four functional requirements were identified.

* FR01 – Retrieve and visualise data from a BLE sensor on a gateway and front-end.
* FR02 – Configure filter algorithms using incremental development.
* FR03 – Distribute sensor data from gateway to server, front-end or other intermediate nodes.
* FR04 – Support dynamic software update to distribute filter algorithms from a client front end to other nodes in an IoT system.

**SECTION IV.**

## **Result**

This section demonstrates how the framework can be applied to monitor physical activity and heart rate based on the AAL use case in Section III.B. A tool for tuning filter algorithms is presented as an example of rapid application development. The result is divided into two subsections: rapid development and distribution of algorithms, and a throughput comparison between a native and a Web Bluetooth solution.

### A. Rapid Development and Distribution of Algorithms

The result of the rapid development and distribution of algorithms is divided into three parts: examples of iterative development of filter algorithms, how to use the framework for rapid development, and finally connection of new sensors to the web gateway.

#### 1) Iterative Development of Filter Algorithms

Fig. 3 shows examples of four common digital filters for streaming data: Exponentially Weighted Moving Average (EWMA), adaptive sampling, zero-crossing rate, and thresholds. Each filter in Fig. 3 was created with the selectors described in Section III. A. Combinations of these filters are also feasible for construction, as exemplified in the simple step counter and the heart rate algorithms shown at the bottom of Fig. 3. The step counter and heart rate algorithms are two examples of filter algorithms developed in an interactive process built on the use case in Section III.B. The first was a step counter, where the sensor was placed in the left jeans pocket. Fig. 4 shows the raw data (black line) and filtered data (red line) from the combined acceleration vector of the triaxial accelerometer. The algorithm was developed in two phases. First, after several iterations, a simple EWMA filter was selected. In the second phase, a threshold value was adjusted to detect a step with the left foot. The speed of the iterations to construct and tune the filter algorithm was increased by the framework. The result of the filter algorithm code in the text field was visualised immediately in the HTML5 canvas on the gateway or front-end, by pressing the <monospace>Program sensor</monospace> button. Fig. 3 shows the final filter for the simple step counter. The number of steps is indicated in Fig. 4 (top left), using the selector <monospace>infoText</monospace>.

Fig. 4. - Contador de pasos simple basado en datos del acelerómetro al caminar.  La línea negra son los datos sin procesar del acelerómetro, y la línea roja son datos filtrados.


**Fig. 4.** Simple step counter based on accelerometer data when walking. The black line is the raw accelerometer data, and the red line is filtered data.

The second example of iterative development is the detection of heart rate, which was retrieved from a pulse oximeter attached to the micro:bit. Here, an iterative approach resulted in an EWMA filter that changed the weight of the moving average depending on whether the value of the received sample was higher or lower than the previous sample. Subsequently, a zero-crossing rate filter was applied to find the intersection crosses between the raw and filter curve to detect a heartbeat. After identifying a heartbeat, the heart rate was calculated and visualised in <monospace>infoText</monospace> as seen in Fig. 5 (top left). The figure shows the raw (in black) and filtered (in red) data from the pulse oximeter. The final filter algorithm to detect the heart rate is seen in Fig. 3.

Fig. 5. - Datos de frecuencia cardíaca de un oxímetro de pulso.  La línea negra son los datos sin procesar del acelerómetro, y la línea roja son los datos filtrados.


**Fig.5.** Datos de frecuencia cardíaca de un oxímetro de pulso. La línea negra son los datos sin procesar del acelerómetro, y la línea roja son los datos filtrados.

#### 2) El marco

El marco presentado es este documento es una herramienta para el desarrollo rápido de aplicaciones para ser utilizado por los desarrolladores de aplicaciones. La puerta de enlace web y el front-end permiten a los desarrolladores de aplicaciones construir y ajustar algoritmos de filtro cuando se introducen nuevos sensores. Los algoritmos de filtro ejemplificados en el contador de pasos y el monitor de frecuencia cardíaca dependen del ruido, la frecuencia de muestreo y la resolución de las muestras entrantes. Estas características pueden cambiar al introducir nuevos sensores, es decir, que requieren la construcción y ajuste de nuevos algoritmos de filtro por parte de un desarrollador de aplicaciones. Este desarrollo paso a paso se puede realizar de forma remota por aire en tiempo real con el marco sin la necesidad de reiniciar el nodo del sensor o reiniciar cualquier parte del sistema. Los datos sin procesar se pueden grabar y guardar en un archivo. Este archivo guardado permite a los desarrolladores probar diferentes algoritmos de filtro en los mismos datos o procesar los datos en otro lugar. Las mediciones de la frecuencia cardíaca y la actividad física se usan con frecuencia en un escenario AAL. Los algoritmos de filtro en la subsección anterior demuestran una prueba de concepto usando el marco para desarrollar aplicaciones AAL que monitorean la actividad física y la frecuencia cardíaca como se describe enSección III.B . Además, un desarrollador de aplicaciones puede integrar fácilmente otros sensores externos en el hogar con el marco.

#### 3) Conexión de nuevos sensores BLE a la Pasarela web

La puerta de enlace web es una parte central del marco y representa dos tercios del tamaño total del código. La puerta de enlace web también resuelve el problema de conectarse a nuevos sensores BLE que requieren la descarga de nuevas aplicaciones. Un sensor puede actuar como una baliza Bluetooth que transmite una URL que se refiere a la puerta de enlace web que se puede configurar explícitamente para el sensor BLE específico. Esta solución permite que un usuario final se conecte a un sensor BLE directamente desde un navegador web y transmita datos.

### B. Comparación de rendimiento entre soluciones Bluetooth nativas y web

Se realizó una prueba de rendimiento para comparar una puerta de enlace web que implementa la API web de Bluetooth con una aplicación desarrollada en Swift que se ejecuta de forma nativa en OS X. Fig. 6muestra el rendimiento para diferentes frecuencias de paquetes. La carga útil de cada paquete era de 12 bytes, ya que es igual al tamaño de la muestra transmitida desde un acelerómetro triaxial. La línea azul representa el máximo teórico. Las siguientes líneas muestran los diferentes resultados para uno o dos sensores, ya sea conectados a un navegador con Bluetooth Web API o una aplicación nativa desarrollada en Swift. La prueba también comparó el rendimiento utilizando notificaciones e indicaciones. La diferencia es que las indicaciones requieren un acuse de recibo por cada paquete enviado. La conclusión fue que a una velocidad de transmisión más alta, una aplicación nativa supera a la aplicación HTML5 Web Bluetooth. La sobrecarga de ejecutar una máquina virtual JavaScript explica la diferencia en el rendimiento. La segunda observación fue el mayor rendimiento cuando se conectaron múltiples sensores. El mayor rendimiento se debió a que el intervalo de solicitud para cada sensor para transmitir datos disminuye de 18 ms a 14 ms. Conectar más de dos sensores no disminuyó aún más el intervalo de tiempo.

[[Fig. 6. - Comparación de rendimiento de sensores BLE que envían paquetes de 12 bytes.
](https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/8502682/8531078/8531124/jonas6-p6-jonas-large.gif)](https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/8502682/8531078/8531124/jonas6-p6-jonas-large.gif)

**Fig.6.** Comparación de rendimiento de sensores BLE que envían paquetes de 12 bytes.

**SECCION V.**

## **Discusión**

En las aplicaciones deportivas y de eSalud, a menudo es necesario realizar múltiples iteraciones durante la construcción y el ajuste de los filtros para obtener la configuración correcta. Este tipo de desarrollo se puede realizar de forma inalámbrica en tiempo real sin reiniciar los nodos ni reiniciar los programas. Además, el marco también funciona sin conexión con datos de sensores almacenados. En educación, proporciona a los estudiantes una herramienta para construir y comprender cómo funcionan los filtros digitales. Este marco se puede utilizar en una solución BLE IoT implementada, ya que la API web Bluetooth también funciona en un nodo.js. Una puerta de enlace node.js también puede actuar como un puente para trabajar en combinación con MQTT y CoAP. Sin embargo, actualmente, el marco está limitado a una puerta de enlace web. Nuestra solución no tiene el mismo rendimiento que una solución nativa como se muestra en la Sección IV.B. Sin embargo, es una solución genérica que es independiente de la plataforma, y ​​las diferencias de rendimiento tienen un rendimiento superior al utilizado habitualmente para los sensores BLE. Se observaron diferencias visibles en el rendimiento a muestras de 100 Hz y superiores con notificaciones ( Fig. 6 ). El desarrollo del marco para admitir una puerta de enlace node.js y varios sensores se considerarán en el trabajo futuro.

**SECCION VI.**

## **Conclusiones**

Este artículo presentó un marco JavaScript para el rápido desarrollo de aplicaciones en sistemas BLE IoT distribuidos, ejemplificado con una herramienta para el desarrollo de algoritmos de filtro. Hemos demostrado que es posible combinar HTML5 Canvas, WebSocket y Bluetooth Web API en un marco para desarrollar, ajustar e implementar iterativamente algoritmos de filtro para aplicaciones de salud electrónica y deportes. El marco puede verse como una parte vital de un sistema de software basado en la web para servicios de salud que utiliza JavaScript y HTML5 para permitir el desarrollo rápido y la actualización dinámica del software en tiempo real para probar diferentes algoritmos dinámicamente sin reiniciar los nodos o reiniciar programas.

### RECONOCIMIENTO

Gracias a Martin Jacobsson por introducir micro: bit y los problemas con los filtros digitales.