ISSN Electrónico: 2500-9338
Volumen 22-N°4
Año 2022
Págs. 154 – 164

# METODOLOGÍA PARA EL DESARROLLO DE PROTOTIPOS ELECTRÓNICOS QUE INTEGREN HARDWARE Y SOFTWARE

Maira Gasca[1]
Enlace ORCID: https://orcid.org/0000-0003-0801-1161

Byron Medina[2]
Enlace ORCID: https://orcid.org/0000-0003-0754-8629

Luis Camargo[3]
Enlace ORCID: https://orcid.org/0000-0002-7956-441X

Fecha de Recepción: 10 de Septiembre 2022
Fecha de Aprobación: 29 de Noviembre 2022

**Resumen**

El artículo presenta una propuesta metodológica para el desarrollo de prototipos que integren hardware y software. Esta propuesta parte de la identificación de las diferencias y similitudes presentes en el desarrollo de firmware y de hardware electrónico, y de las complicaciones de su integración en un prototipo. La propuesta se fundamenta en los consejos de la programación pragmática y los manifiestos de las metodologías ágiles y cíclicas para el desarrollo de software, la estrategia Bottom-Up utilizado para el desarrollo de hardware y algunas de las actividades de la metodología Out-In. La metodología está divida en fases y etapas. La fase de inicialización comprende las etapas de relato, análisis y bala trazadora. La fase de desarrollo es un sprint que inicia con la bala trazadora, posteriormente pasa por varios sprint de la etapa de construcción, luego la etapa de integración, y finaliza con la etapa de pruebas de laboratorio. La etapa de verificación integra la etapa de demostración y entrega; desde esta fase se puede regresar a la fase de desarrollo si no se cumplen los requerimientos del prototipo y se vuelve a iterar. La metodología fue aplicada en la elaboración de un prototipo, obteniendo resultados satisfactorios en el cumplimiento de los requerimientos y en el tiempo de ejecución. Se concluye que es posible obtener un prototipo factible, que integre hardware y software con un nivel de madurez TRL6, en un tiempo específico, usando la metodología propuesta en un entorno trabajo colaborativo e iterativo.

**Palabras claves:** Firmware, Electrónica, Metodologías Ágiles, Bottom-Up, Iteración**.**

[1] Doctora en Ciencias, Universidad Rafael Belloso Chacín; Universidad Antonio Nariño, Colombia, Contacto: mairacgm@gmail.com; magasca@uan.edu.co
[2] Doctor en Ciencias, Universidad Rafael Belloso Chacín; Universidad Francisco de Paula Santander, Colombia, Contacto: byronmedina@ufps.edu.co
[3] Doctor en Ciencias, Universidad Rafael Belloso Chacín; Universidad del Magdalena, Colombia, Contacto: lcamargoa@unimagdalena.edu.co

## METHODOLOGY FOR THE DEVELOPMENT OF ELECTRONIC PROTOTYPES INTEGRATING HARDWARE AND SOFTWARE

**Abstract:**

The article presents a methodological proposal for developing prototypes that integrate hardware and software. This proposal starts with the identifying the differences and similarities presented in the development of firmware and electronic hardware, and the complications of their integration into a prototype. It is based upon insights from pragmatic programming, agile and cyclical methodologies for software development, the Bottom-Up strategy used in hardware development, and some activities from the Out-In methodology. The methodology is divided into phases and stages. The initialization phase comprises the stages of story, analysis, and tracer bullet. The development phase consists of a sprint that commences with the tracer bullet, followed by several sprints of the construction stage, then the integration stage, and culminates with the laboratory testing stage. The verification stage combines the demonstration and delivery stage. If the prototype requirements are not met, one can return to the development phase and iterate again. The methodology was applied in the development of a prototype, yielding satisfactory results in meeting the requirements and in execution time. It is concluded that it is feasible to obtain a prototype integrating hardware and software at a TRL6 maturity level within a specific timeframe, through the proposed methodology in a collaborative and iterative work environment.

**Keywords:** Firmware, Electronics, Agile Methodologies, Bottom-Up, Sprint.

## METODOLOGIA PARA O DESENVOLVIMENTO DE PROTÓTIPOS ELETRÔNICOS INTEGRANDO HARDWARE E SOFTWARE

**Resumo:**

O artigo apresenta uma proposta metodológica para o desenvolvimento de protótipos que integram hardware e software. Esta proposta começa com a identificação das diferenças e semelhanças presentes no desenvolvimento de firmware e hardware eletrônico, bem como as complicações de sua integração em um protótipo. Baseia-se em insights da programação pragmática, metodologias ágeis e cíclicas para o desenvolvimento de software, na estratégia Bottom-Up utilizada no desenvolvimento de hardware e em algumas atividades da metodologia Out-In. A metodologia é dividida em fases e etapas. A fase de inicialização compreende os estágios de história, análise e marcador. A fase de desenvolvimento consiste em um sprint que começa com o marcador traçador, seguido por vários sprints da fase de construção, depois a fase de integração e culmina com a fase de testes em laboratório. A fase de verificação combina a fase de demonstração e entrega. Se os requisitos do protótipo não forem atendidos, pode-se retornar à fase de desenvolvimento e iterar novamente. A metodologia foi aplicada no desenvolvimento de um protótipo, obtendo resultados satisfatórios no atendimento aos requisitos e no tempo de execução. Concluise que é viável obter um protótipo integrando hardware e software no nível de maturidade TRL6 dentro de um prazo específico, através da metodologia proposta em um ambiente de trabalho colaborativo e iterativo.

**Palabras chave:** Firmware, Eletrônica, Metodologias Ágeis, Bottom-Up, Iteração.

## 1. INTRODUCCIÓN

Un prototipo hace referencia a una máquina de prueba que tiene características semejantes al sistema deseado. El prototipo es usado para valorar el desempeño de la tecnología desarrollada antes de su producción y comercialización. La escala de nivel de preparación del prototipo permite identificar la madurez de las tecnologías durante su desarrollo. Para que un desarrollo se convierta en prototipo viable debe superar pruebas de factibilidad en condiciones de operación, alcanzando como mínimo un nivel de madurez TRL 6 (TRL, *Technology Readiness Levels*) (Mankins, 1995).

Los prototipos electrónicos son máquina formadas en su mayoría por componentes electrónicos, aunque puede contener elementos: eléctricos, mecánicos, electromecánicos, neumáticos, electroquímicos, entre otros. Estos elementos se interconectan e interactúan para cumplir las funciones del prototipo. Los prototipos electrónicos contienen sub-sistemas como: sensado, actuadores, procesamiento, almacenamiento, comunicaciones, soporte energético, entre otros (Medina, Castro, & Camargo, 2015).

Algunos subsistemas están compuestos por hardware y software; estos cuentan con algún componente electrónico que tiene la capacidad y el requerimiento de ejecutar un algoritmo computacional para desarrollar correctamente su función dentro del sistema. Adicionalmente, la generalidad de los prototipos electrónicos modernos cuenta con una unidad de procesamiento donde se gobiernan los subsistemas y se encuentra la rutina principal del prototipo (Castro, Medina, & Camargo, 2016). Esta unidad puede estar compuesta por un microcontrolador o un microprocesador y algunos periféricos integrados en una placa de desarrollo o en un solo circuito integrado (SoC*, System-on-a-Chip*) como: Arduino, Raspberry, Jetson, ESP32, Intel Arria, Intel Edison, entre otras soluciones tecnológicas semejantes.

El programa o soporte lógico que controla los componentes electrónicos es llamado firmware. Este es el encargado de configurar, integrar, coordinar y controlar el hardware específico de una máquina electrónica, asegurando su correcto funcionamiento. El firmware es inalterable y siempre está embebido en la electrónica, lo que implica que no puede ser instalado y desinstalado por el usuario. Pero, este no deja de ser un programa informático; así que, el diseño y codificación del firmware es uno de los puntos de unión entre el hardware y software, en el desarrollo de prototipos (Davidson & Bruce, 1978).

El desarrollo de firmware trae consigo nuevos retos que las metodologías para el desarrollo de software tradicional no satisfacen completamente; porque este se ejecuta en un hardware diferente a los computadores, tabletas o teléfonos inteligentes. El desarrollo, la simulación, la emulación y las pruebas normalmente se realizan en un hardware más robusto (computador) que el hardware donde se va a ejecutar. Esto causa algo de incertidumbre en: la integridad, la seguridad y en los tiempos de ejecución (Wright, Moeglein, Bagchi, Kulkarni, & Clements, 2021). Adicionalmente, el firmware debe funcionar con una tasa de ocurrencia de fallas muy baja, teniendo en cuenta que es el directo responsable del proceso, y si él falla la máquina también falla. La actualización, parcheo o corrección de un firmware se realiza con menor frecuencia que un programa para computador; considerando que en la mayoría de las veces estos procesos implican una re-factorización (Tieling Zhang, 2005). Por esto, el modelado de confiabilidad operacional del firmware debe basarse en las características de falla de la electrónica y el firmware, y no en las de un software tradicional (Gasca, Camargo, & Medina, Gestión del mantenimiento para la confiabilidad operacional, 2020). El enfoque tradicional se limita a analizar las fallas en la fase de desarrollo y no contempla las fallas durante la operación, en donde predominan las fallas del hardware (Tieling Zhang, 2005).

Por otra parte, la elaboración de un prototipo de hardware tiene un alto costo en comparación con el desarrollo de software tradicional. Esto se debe a que en la construcción se adquieren y se desechan piezas que no hacen parte del producto final; además, el diseño, la implementación y las pruebas en un entorno real de un hardware tienen un costo más alto que el requerido por un software (Thompson, 2015). Aunque, esto se está reduciendo con el uso de herramientas computacionales (CAD*, Computer Aided Design*) para simular y verificar circuitos electrónicos (Schweers, 2002); sin embargo, estas aun no emulan con precisión todas las características del entorno de uso y de los elementos utilizados.

Existen estrategias para la elaboración de hardware enfocadas en la descripción del orden como se desarrolla el producto. Estas son usadas comúnmente en el hardware descrito.

La estrategia de abajo hacia arriba (*Bottom-up*) consiste en integrar componentes para formar módulos o sub-sistemas y luego interconectarlos para formar el sistema o producto. Esto requiere de mucha experiencia en el equipo desarrollador, teniendo en cuenta que tiene que visionar el producto final; además, se presenta dificultad al conseguir y corregir un error, cuando el sistema está compuesto por muchos elementos y estos están poco especificados (McFarland, 1988).

La estrategia de arriba hacia abajo (*Top-down*) parte de una idea abstracta del producto y esta se va implementando en diferentes niveles de detalle hasta lograr las especificaciones del producto final; para esto, el sistema se divide en módulos y éstos en sub-módulos de una forma jerárquica; esta división se realiza las veces que sea requerida hasta los elementos primarios. Esta estrategia obtiene más rápido los primeros prototipos, aunque no sean completamente funcionales; pero requiere de mayor tiempo, planeación, colaboración y documentación, para evitar conflictos de diseño entre las entidades y la pérdida en el desempeño a nivel de entidad y del sistema general (Kaeslin, 2014). En algunos casos es difícil alcanzar la idea inicial porque no se consideran las limitaciones particulares de cada elemento.

Estas estrategias se han intentado fusionar en metodologías para el desarrollo de hardware y software, como es el método en V, en donde las especificaciones, el modelado, el diseño de detalle y la implementación de la unidad se realizan de arriba hacia abajo; y las pruebas de unidad, la integración y pruebas del sistema se realizan de abajo hacia arriba (Perez, Berreteaga, Ruiz de Olano, Urkidi, & Perez, 2006). El método V pretende obtener lo más favorable de *Top-down* y *Bottom-up*. Pero, esta metodología se estructura siguiendo las metodologías tradicionales con mucha documentación y rigidez que no admiten realizar cambios ágiles y útiles durante el desarrollo; además, no contempla el trabajo iterativo y repetitivo requerido para obtener un producto de calidad a través de diferentes versiones de este.

El enfoque de diseño inverso orientado a objetivos (*GoID*) también ha sido utilizado para el desarrollo de este tipo de productos. Este parte de la identificación de resultados esperados o funciones del prototipo, y a partir de estas se planifican los objetivos, las actividades y la secuencia de fabricación (*Top-down*). Aunque la visión del prototipo no es tan abstracta, si se toman decisiones equivocadas en el proceso, se generan conflictos de diseño entre las entidades y la pérdida en el desempeño a nivel de entidad y del sistema general (Mathew & Anand, 2022).

El modelo de desarrollo en espiral unido al trabajo colaborativo con la academia facilita la obtención de productos de excelente calidad respaldados por el proceso de investigación de las universidades y, la publicación de artículos y libros científicos arbitrados sobre el detalle de la tecnología utilizada en el prototipo (Mendoza, Serrano, & Luz, 2022). Pero, esta metodología en la mayoría de los casos no ajusta el tiempo de desarrollo con el tiempo requerido para sacar el producto al mercado antes de perder la innovación.

La metodología de afuera hacia dentro (OIM, *Out–In Methodology*) utilizada para el desarrollo de prototipos presenta un enfoque basado en la confiabilidad del firmware; para esto se codifica la funcionalidad principal junto con una sencilla interfaz para identificación de errores y pruebas (Subramanian & Lawrence, 2000). Este método no contempla en detalle temas relacionados con el desarrollo de hardware.

Teniendo cuenta que las estrategias o metodologías mencionados no suplen completamente los requerimientos e intenciones del desarrollo de prototipos hardware y software, se plantea la siguiente pregunta de investigación:

¿Qué metodología se debe seguir para desarrollar en un tiempo prudente un prototipo con hardware y firmware factible?

En este artículo se propone una metodología acorde con el desarrollo de un prototipo electrónico. El objetivo principal de la metodología es facilitar la creación e implementación de hardware y software confiable en un producto tecnológico que cumpla los estándares del nivel de madurez tecnológico TRL 6 en el menor tiempo posible. Esta metodología es utilizada y validada en el proceso de construcción de un prototipo llamado NodoMóvilSensor.

El prototipo NodoMóvilSensor mide, almacena y envía los datos de las siguientes variables ambientales: precipitación, PM25, PM10, CO2, radiación solar, temperatura, presión atmosférica y viento; mientras éste se mueve por la ciudad.

## 2. METODOLOGÍA

Para el desarrollo de la propuesta primero se realiza una revisión bibliográfica, con el propósito de identificar y adoptar los fundamentos teóricos que se aplican en esta metodología. A continuación, se describen las fases de la metodología y las interacciones y relaciones de éstas dentro de un marco de estrategia global que guía y optimiza el proceso y el producto. Posteriormente, se aplica el método propuesto para la elaboración de prototipos que integran hardware y software.

### Fundamentos teóricos de la propuesta

La metodología propuesta se fundamenta en: la programación pragmática y las metodologías ágiles y cíclicas para el desarrollo de software, la estrategia Bottom Up utilizada para el desarrollo de hardware y algunas de las propuestas de la metodología Out-In.

De las metodologías ágiles y cíclicas se heredan los valores de simplicidad, flexibilidad, desarrollo iterativo e incremental, y continua comunicación con el cliente (Gasca, Camargo, & Medina, Metodología para el desarrollo de aplicaciones móviles, 2014). Estos valores están enmarcados en los cuatro postulados del manifiesto ágil.

- Desarrollar software que funciona más que conseguir buena documentación (Beck, y otros, 2001).
- La respuesta ante el cambio es más importante que el seguimiento de un plan (Beck, y otros, 2001).
- Colaboración con el cliente y con el equipo desarrollador, sobre negociación contractual o documentación inicial (Beck, y otros, 2001).
- Individuos e interacciones sobre procesos y herramientas (Beck, y otros, 2001).

Aplicando estos valores en el proceso, se espera reducir el tiempo de construcción del prototipo e incrementar el trabajo colaborativo en el personal involucrado.

De la programación pragmática se emplean los consejos para la interpretación y análisis de los problemas y fallas presentadas en el prototipo durante el proceso de construcción. Algunos de estos consejos se mencionan a continuación:

- Evitar mal olor o ventanas rotas *(a bad smell*), si no se corrige un mal desempeño del prototipo en un corto plazo, los demás miembros del equipo desarrollado dejarán de preocuparse por el error y el prototipo declinará. Se debe re-factorizar en el menor tiempo posible ante fallos leves (Thomas & Andrew, 2019).
- No se re-factoriza y se agregan nuevas funcionalidades al prototipo al mismo tiempo, y si la refactorización es grande se debe hacer en pequeños pasos, para evitar el desacople entre los diferentes componentes, debido a que toda re-factorización genera cambios en el producto (Thomas & Andrew, 2019) (Bendix & Torbjörn, 2009).
- El primer prototipo solo sirve para aprender. Si el experimento no funciona se buscan nuevas formas de hacerlo y si funciona se re-diseña para mejorar (Thomas & Andrew, 2019).
- El uso de balas trazadoras (*Tracer Bullets*). (Thomas & Andrew, 2019).

La aplicación de estos consejos en el proceso facilita la construcción de un prototipo confiable, porque se pueden reducir las posibles fallas al depurar el prototipo en el menor tiempo posible. Además, la aplicación del consejo sobre el uso de la bala trazadora obliga al equipo a desarrollar una versión simple y desechable del prototipo que se pueda mostrar al cliente en un corto tiempo; y así, si esa bala da en el blanco los futuros prototipos también.

La metodología propuesta parte de la etapa de conceptualización del prototipo, en donde se plantean los objetivos y la bala trazadora con base en los requerimientos de éste; luego, se desglosan sus partes con sus especificaciones técnicas, siguiendo un orden y jerarquía descendente. Sin embargo, la implementación de las partes, componentes y la integración de éstos no sigue la estrategia *Top-down,* sino *Bottom-up*.

La incorporación de la estrategia *Bottom-up* en la metodología propuesta, facilita la implementación, porque se parte de algo pequeño y simple, que va creciendo a medida que se van desarrollando y reuniendo los diferentes componentes para formar los subsistemas y el producto final. Esto permite que cada componente se realiza mediante iteraciones con reglas, procesos y tiempos específicos; esta espiral es válida y única para cada componente (*Ad hoc*). Una iteración contiene uno o varios procesos, entre los cuales pueden estar: diseño, simulación, análisis, síntesis, construcción, validación e integración, entre otros. El *Ad hoc* se debe a que un componente contiene hardware y/o software con diferentes requerimientos, complicaciones y limitaciones.

De OIM (Subramanian & Lawrence, 2000) se hereda la construcción y el uso de un software de interfaz para pruebas y validación de cada parte del prototipo. Esta interfaz facilita la visualización de los resultados de forma simple y evita en la mayoría de los casos el uso de instrumento de medición. Además, ésta permite observar la evolución de la implementación y la integración en un entorno agradable para todo el equipo de trabajo.

**Descripción de la propuesta metodológica**

Esta metodología requiere del trabajo colaborativo e iterativo y la incorporación de procesos en serie, paralelo, síncronos y asíncronos.

La metodología se enmarca en tres grandes fases: inicialización, desarrollo y verificación. En la figura 1 se muestran las fases de la metodología propuesta.

**Figura 1.** Fases y etapas de la metodología

En la fase de inicialización se realizan las siguientes etapas:

- Relato: en esta etapa se realizan las reuniones entre el cliente o interesado y el equipo de trabajo; para identificar los hechos que describen el problema y las posibles soluciones que ofrecerá el prototipo.
- Análisis: en esta etapa se plantean los objetivos del prototipo, se definen el rumbo y las metas a cumplir para la obtención del producto final, y se identifican los requerimientos funcionales y no funcionales del prototipo.
- Bala trazadora: en esta etapa se realiza la descripción comportamental y estructural del prototipo, se descompone el prototipo en sus diferentes partes, y se asignan tareas y roles dentro del equipo de trabajo.

La fase de desarrollo inicia con la bala trazadora, y contiene las etapas de: construcción, integración y pruebas de laboratorio.

La fase de desarrollo comprende varios ciclos o iteraciones (*sprints*). Cada ciclo representa la elaboración e integración de un elemento, sub-sistema o sistema dentro del producto final. Estos *sprints* pueden desarrollase en paralelo o serie, pero siempre la complejidad del producto aumenta con cada *sprints*. Esta fase inicia con el desarrollo de elementos, posteriormente sub-sistemas, luego sistemas, y así hasta alcanzar el prototipo final (*Bottom-up)*.

- Construcción: la etapa de construcción también se debe llevar a cabo en iteraciones o *sprints*; estas pueden ser para construir: solo hardware, solo software, o hardware y software. Cuando el componente a implementar requiere de hardware y software se deben desarrollar en paralelo los *sprints* (hardware y software); y estos se deben alinear en el límite del primer *sprints*. Por ejemplo: se requieren dos *sprint* para el software y un *sprint* para el hardware, y se alinean al terminar el ciclo del hardware; o se requieren dos *sprint* para el hardware y un *sprint* para el software, y se alinean al terminar el ciclo de software. Adicionalmente, cada ciclo de esta etapa tiene reglas, procedimientos y limitaciones propias del componente a construir.
- Integración: en esta etapa se conecta el componente del prototipo elaborado en la etapa de construcción con el producto de la iteración de la fase de desarrollo anterior. En la integración se necesita que los componentes cumplan con las especificaciones de diseño (formato de datos, bus de comunicación, puerto, niveles de energía, número de terminales de conexión, entre otros); además, para facilitar la integración se utiliza la interfaz gráfica en la prueba del prototipo.
- Pruebas de laboratorio: en esta etapa se valida el funcionamiento de cada componente en el laboratorio con el uso de la interfaz gráfica para prueba y equipos de medición. En el último sprint de la fase de desarrollo se valida el prototipo completo.

La fase de verificación inicia cuando en la fase anterior se logra validar el prototipo completo en un laboratorio, y finaliza cuando el prototipo es evaluado satisfactoriamente en condiciones de operación del entorno real (demostración superada). Si los resultados de la demostración en el entorno real no son satisfactorios se debe regresar a la fase de desarrollo o a la fase de inicialización según la gravedad y complejidad del problema.

- Demostración: en esta etapa se prueba el prototipo en condiciones de operación real. El prototipo se instala y se pone en funcionamiento durante un tiempo, en donde se puedan presentar las posibles condiciones de operación y del ambiente. En este tiempo se supervisa el desempeño del producto con el propósito de registrar, evaluar y aprender sobre la tecnología desarrollada. Adicionalmente, de esta etapa pueden surgir las políticas de mantenimiento preventivo y correctivo del prototipo.

Terminada la depuración de la aplicación y atendidos todos los requerimientos de última hora, se da por finalizado el desarrollo del prototipo y se procede a la

entrega del prototipo al cliente, para que él continúe con el proceso de certificación TRL 7 y TRL 8.

- Entrega: en esta etapa se documentan y se entregan los ejecutables, código fuente, planos, listas de componentes, diagramas o representaciones del sistema, registro de los resultados de las pruebas y el manual del sistema.

## 3. RESULTADOS

Resume los hallazgos relacionados por cada uno de los objetivos planteados, presenta las propias observaciones con otros estudios de interés y, señala las aportaciones y limitaciones de unos y otros. Se mencionan las limitaciones que se tuvieron en la investigación, incluyendo las deducciones para una investigación futura.

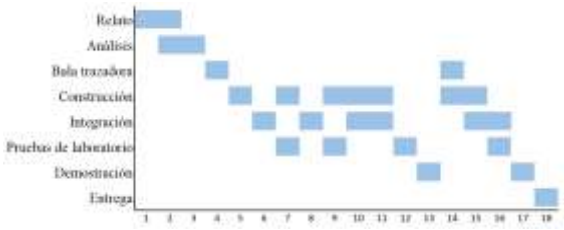
El método elaborado se aplica para el desarrollo de un prototipo que integra hardware especializado y software.

El prototipo tiene por nombre NodoMóvilSensor es uno de los productos obtenidos en el proyecto de investigación titulado “Desarrollo de un nodo móvil sensor y un nodo sumidero fijo para una red de sensores inalámbrica en la ciudad de Santa Marta. Proyectando una ciudad inteligente.” Este proyecto fue desarrollado por un equipo de profesores y estudiantes de dos universidades de Colombia.

El prototipo fue dividido comportamental y estructuralmente en: unida central de procesamiento, comunicación, medición de precipitación, medición de PM25 y PM10, medición de CO2, medición de radiación solar, medición de temperatura, medición de presión atmosférica y medición de viento. En estas partes se enfatizó en el requerimiento principal que era la movilidad.

En la figura 2 se muestra la ejecución de las diferentes etapas de la metodología y su dedicación de tiempo en semanas.

**Figura 2.** Tiempo de ejecución semanal por etapa.

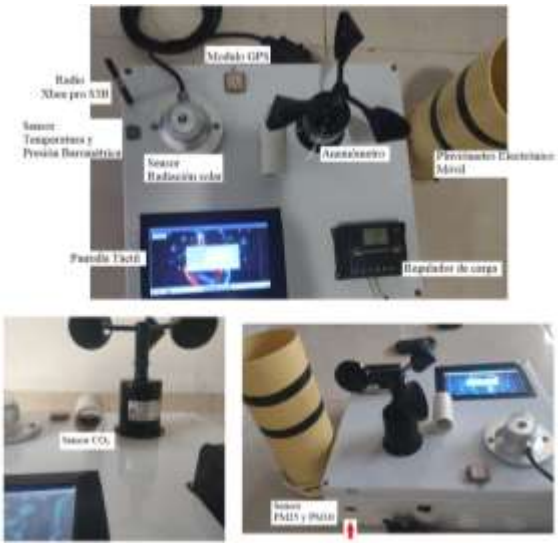



A continuación, se describen el hardware y software obtenido y el resultado de las pruebas del prototipo.

**Descripción del hardware**

El hardware del NodoMóvilSensor se compone de: un radio de largo alcance Digi Xbee RF, una computadora de placa reducida Raspberry pi, una pantalla de 7 pulgadas táctil, un teclado inalámbrico, una batería sellada, un módulo solar policristalino con potencia de 20 Wp a 12 Vdc, un regulador de carga, varios convertidores analógicos a digital de 16 Bits con comunicación I2C para los sensores con salida analógica, y varios convertidores UART a USB para los sensores con salida digital. Los sensores conectados al bus I2C de la Raspberry son: MPL115A2 (temperatura y presión barométrica), Pluviómetro Electrónico Móvil (precipitación), Atlas Scientific EZO-CO2 (CO2), RK100-02 (anemómetro) y RK200-04 (radiación solar). Los dispositivos conectados al bus USB de la Raspberry son: el radio Xbee pro S3B (900 MHz), el sensor HPMA115SO Honeywell (PM25 y PM10), sensor NEO M8M (módulo GPS), el táctil de la pantalla y el terminal del teclado inalámbrico. En la figura 3 se muestra el exterior del hardware del prototipo elaborado.

**Figura 3.** Partes del hardware del prototipo.




**Descripción del Software**

El firmware del NodoMóvilSensor es un conjunto de programas que controla los circuitos y dispositivos electrónicos del prototipo. Este software fue codificado en Phyton 3 y desarrollado para ser ejecutado en un computador de placa reducida Raspberry pi con sistema operativo GNU/Linux basado en Debian (Raspberry Pi OS).

Este software tiene como funciones principales: a). Permitir la configuración de los radios Xbee. b). La revisión de la conexión de los sensores. c). La configuración, linealización e instrumentación de algunos sensores d). La medición de las variables ambientales georreferenciada, temporizada y promediada en una cuadrícula espacial de 2500 metros cuadrados (cada vez que el nodo sensitivo se desplaza 50 metros.) e). La composición y el almacenamiento de la trama de la medición. f). La comunicación con el nodo sumidero y envío del archivo con las tramas de información. g). La visualización de la actividad del nodo sensor móvil (medición y envío).

En la figura 4 se presentan algunas pantallas de la interfaz del prototipo, en donde se muestra y valida el funcionamiento de este.

**Figura 4.** Pantallas de la interfaz del firmware del prototipo.


**Descripción de la verificación**

Siguiendo la metodología se realizan pruebas de unidad, pruebas finales del prototipo en un ambiente controlado y en un entorno real de funcionamiento.

Las pruebas de unidad se desarrollaron a medida que se realizaba cada elemento, subsistema o sistema del prototipo. Estas pruebas facilitaron la linealización e instrumentación de los sensores y la integración de los componentes al producto final.

Las pruebas de funcionamiento del prototipo se realizaron una vez terminado todo el sistema propuesto. Sistema que incluye NodoMóvilSensor, NodoSumidero y el Servidor, y que se muestra en la figura 5.

**Figura 5.** Sistema completo para medición ambiental.

El prototipo NodoMóvilSensor funcionó de forma correcta cuando se integró con las demás partes del proyecto, validando cada una de sus funciones. Esta integración y prueba se realiza en el laboratorio monitorizando las entradas y las salidas.

Superada la evaluación en el laboratorio se realiza la demostración del prototipo en condiciones de operación real. El NodoMóvilSensor se coloca en movimiento; el NodoSumidero en una intersección vehicular, y el servidor en Internet. En la figura 6, se muestra el prototipo NodoMóvilSensor instalado.

**Figura 6.** Prototipo instalado en un vehículo para demostración.

Los resultados de las diferentes pruebas realizadas determinaron la factibilidad y el TRL6 del prototipo. El error cuadrático medio de las medidas ambientales es inferior 3% con respecto a las mediciones realizadas por sensores estacionarios.

## 4. DISCUSIÓN Y CONCLUSIONES

Es posible obtener un prototipo factible que integra hardware y software con un nivel de madurez TRL 6, en un tiempo específico, usando la metodología propuesta en un entorno de trabajo colaborativo e iterativo.

El desarrollo de firmware y el desarrollo de hardware electrónico tienen aspectos en común y grandes diferencias que fueron analizados e integrados satisfactoriamente en la metodología propuesta. El uso de condiciones específicas en cada iteración para la construcción de cada componente del prototipo, en la etapa de desarrollo, facilitó la integración y sincronización entre el hardware y software. Adicionalmente, el uso de procesos seriales y en paralelo, propuestos en la metodología, acortó los tiempos de desarrollo de prototipos de software y hardware.

En la elaboración del prototipo donde se validó la metodología propuesta, el requerimiento más difícil de cumplir fue el de movilidad en el sensado. En la primera iteración de la fase verificación, en la etapa de demostración no se cumplió con el objetivo; por esto, se tuvo que regresar a la etapa de bala trazadora y realizar algunos *sprint*s en la fase de desarrollo, y nuevamente realizar la demostración, cumpliendo con el requerimiento. Esto demuestra las ventajas del uso de *sprints* y el trabajo colaborativo en la metodología.

## 5. REFERENCIAS

Beck, K., Beedle, M. B., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., . . . Thomas, D. (2001). Manifesto for agile software development. Utah: The Agile Alliance. https://athena.ecs.csus.edu/~buckley/CSc191/XP.pdf

Bendix, L., & Torbjörn, E. (2009). Software Configuration Management in Agile Development. En Software Applications: Concepts, Methodologies, Tools, and Applications (págs. 291-308). Pensilvania: Igi Global. https://www.igi-global.com/chapter/software-applications-concepts-methodologies-tools/29394

Castro, S., Medina, B., & Camargo, L. (2016). Supervisión y Control Industrial a través de Teléfonos Inteligentes usando un computador de placa única Raspberry Pi. Información tecnológica, 27(2), 121-130. https://doi.org/10.4067/S0718-07642016000200015

Davidson, S., & Bruce, S. (1978). An overview of firmware engineering. Computer, 11(5), 21-33. https://doi.org/10.1109/C-M.1978.218180

Gasca, M., Camargo, L., & Medina, B. (2014). Metodología para el desarrollo de aplicaciones móviles. Tecnura, 18(40), 20-35.

http://www.scielo.org.co/scielo.php?pid=S0123-921X2014000200003&script=sci_arttext

Gasca, M., Camargo, L., & Medina, B. (2020). Gestión del mantenimiento para la confiabilidad operacional. Espacios, 41(47), 250-261. https://doi.org/10.48082/espacios-a20v41n47p18

Kaeslin, H. (2014). Top-down digital VLSI design: from architectures to gate-level circuits and FPGAs. Burlington: Morgan Kaufmann.

Mankins, J. (1995). Technology readiness levels. Washington D. C: NASA. Obtenido de https://aiaa.kavi.com/apps/group_public/download.php/2212/TRLs_MankinsPaper_1995.pdf

Mathew, B., & Anand, B. N. (2022). An Information-Decision Framework to Support Cooperative Decision Making in the Top-Down Design of Cyber-Physical-Manufacturing Systems. ASME 2022, International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. St. Louis. https://doi.org/10.1115/DETC2022-90836

McFarland, M. (1988). Using bottom-up design techniques in the synthesis of digital hardware from abstract behavioral descriptions. Twenty-five years of electronic design automation (págs. 602-608). New York: IEEE. https://doi.org/10.1145/62882.62955

Medina, B., Castro, S., & Camargo, L. (2015). Tecnologías de código abierto para la gestión de un proceso industrial. Revista Gti, 14(38), 43-58. Obtenido de https://revistas.uis.edu.co/index.php/revistagti/article/view/4941

Mendoza, C., Serrano, J., & Luz, R. (2022). La generación de prototipos electrónicos para incrementar la productividad académica. Revista Iberoamericana de Producción Académica y Gestión Educativa, 8(15), 108-130. Obtenido de https://www.pag.org.mx/index.php/PAG/article/view/894

Perez, A., Berreteaga, O., Ruiz de Olano, A., Urkidi, A., & Perez, J. (2006). Una metodología para el desarrollo de hardware y software embebidos en sistemas críticos de seguridad. SISTEMAS, CIBERNÉTICA E INFORMÁTICA, 3(2), 70-75. Obtenido de https://www.iiisci.org/journal/pdv/risci/pdfs/c863gm.pdf

Schweers, R. (2002). Descripción en VHDL de arquitecturas para implementar el algoritmo CORDIC. Tesis Doctoral. Buenos Aires: Universidad Nacional de La Plata. Obtenido de https://sedici.unlp.edu.ar/bitstream/handle/10915/3835/Documento_completo.pdf?sequence=15

Subramanian, N., & Lawrence, C. (2000). Testable embedded system firmware development: the out–in methodology. Computer Standards & Interfaces, 22(5), 337-352. https://doi.org/10.1016/S0920-5489(00)00054-4

Thomas, D., & Andrew, H. (2019). The pragmatic programmer. Reading: Addison-Wesley Professional.

Thompson, K. (2015). Agile Processes for Hardware Development. San Mateo: Cprime, Inc. Obtenido de https://www.cprime.com/wp-content/uploads/2020/10/Agile_Processes_for_Hardware_Development.pdf

Tieling Zhang, M. X. (2005). Reliability and modeling of systems integrated with firmware and hardware. International Journal of Reliability. Quality and Safety Engineering, 12(3), 227-239. https://doi.org/10.1142/S021853930500180X

Wright, C., Moeglein, W., Bagchi, S., Kulkarni, M., & Clements, A. (2021). Challenges in firmware re-hosting, emulation, and analysis. ACM Computing Surveys, 54(1), 1-36. https://doi.org/10.1145/3423167