Plataforma embebida para el apoyo de la atención paramédica en vehículos de transporte asistencial básico

Maria Juliana Garzón Vargas

Trabajo de grado para optar al título de Ingeniera de Sistemas e Informática

Director:

Gabriel Rodrigo Pedraza Ferreira

Doctor en Informática

Codirector:

Carlos Jaime Barrios Hernández

Doctor en Informática

Universidad Industrial de Santander

Facultad Ingenierías Fisicomecánicas

Escuela de Ingeniería de Sistemas e Informática

Bucaramanga

2017

A Dios, mis papás, mi hermanita, mis tías y tíos.

Agradecimientos

Al Centro de Supercomputación y Cálculo Científico (SC3). Al Grupo de Cómputo Avanzado y a Gran Escala (CAGE). Al Director de este proyecto, profesor Gabriel Pedraza Ferreira por su excelencia profesional y humana. Al Codirector, profesor Carlos Jaime Barrios Hernández, por su liderazgo y motivación. A Sergio Augusto Gélvez, Luis Alejandro Torres, Gilberto Díaz, Wanda Catalina Rincón, por su ayuda y orientación incondicional.

Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

Pág.

Resumen

Título: PLATAFORMA EMBEBIDA PARA EL APOYO DE LA ATENCIÓN PARAMÉDICA EN VEHÍCULOS DE TRANSPORTE ASISTENCIAL BÁSICO.[*]

Autor: Maria Juliana Garzón Vargas[**]

Palabras clave: Supercomputación, Arquitectura HW/SW, Asistencia prehospitalaria, Transporte Asistencial Básico, Sistemas embebidos.

Resumen:

La supercomputación es la utilización de técnicas sofisticadas en hardware y software para acelerar la ejecución de programas. La importancia de los supercomputadores se basa en el cálculo numérico a gran escala, tanto en volumen de datos como a nivel de procesamiento. En la actualidad, la tecnología que se encuentra en auge para cómputo avanzado son los sistemas embebidos, su aplicación ha madurado hasta llegar a abordar casi cualquier ámbito de actividad. La asistencia prehospitalaria no es la excepción, la visualización, procesamiento de imágenes, adquisición y transmisión de datos se realiza a partir de equipos biomédicos cuyas funcionalidades están siendo simuladas por este tipo de sistemas. Uno de los principales retos que enfrentan los proveedores de servicios prehospitalarios es la adquisición de estos equipos para la dotación de ambulancias. La asistencia que se efectúa durante el transporte de pacientes exige para la dotación de los vehículos equipos eficientes, precisos y duraderos. Sin embargo, por la naturaleza compleja de sus funcionalidades demandan un alto costo en términos de compra y mantenimiento. El enfoque de este estudio aprovecha la capacidad de cómputo y la eficiencia energética de un supercomputador en un sistema embebido en el que se diseña y construye la arquitectura software y el prototipo funcional que permiten en conjunto la ejecución de aplicaciones avanzadas y genéricas, para así complementar la atención paramédica en ambulancias.

[*] Trabajo de Grado

[**] Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director PhD. Gabriel Pedraza Ferreira. Co-Director PhD. Carlos Jaime Barrios Hernández

Abstract

Title: High performance embedded computing platform for emergency vehicle transportation.[*]

Author: Maria Juliana Garzón Vargas[**]

Keywords: High Performance Computing, HW/SW Architecture, Prehospital care, Basic Life Support, Embedded Systems.

Abstract:

High performance computing is using sophisticated hardware and software techniques to accelerate the execution of programs. The importance of supercomputers is based on the large-scale numerical operations performed with both data volume and processing level. Nowadays, embedded systems are the ultimate technology tool for advanced computing, its application has growth to the point where almost any activity field can be taken into account. Prehospital care emergency services are not exception, either by visualizing, image processing, acquiring or transferring data, biomedical equipments are key answer, and some of its functionalities are simulated by embedded systems. One of the challenges emergency medical services providers face is having this equipments as part of the supplies of ambulances. The patient assistance achieved along the way demands efficiency, accuracy and long-lasting technological equipment inside the ambulance. However, because of the complex nature of its functionalities these devices turn out to be high-cost purchase and maintainance.

The approach of this study seizes the computing capability and power efficiency of a HPC embbeded computer system where the software architecture model and functional prototype are designed and built allowing together the implementation of advanced and generic applications to improve the paramedic assistance offered by ambulances. The ultimate goal to achieve is to support the current service by reducing high costs, power consumption yet keeping reliability results.

[*] Undergraduate Final Project

[**] Physico-Mechanical Engineering Faculty. Systems Engineering and Computer Science School. Adviser PhD. Gabriel Pedraza Ferreira. Co-Adviser PhD. Carlos Jaime Barrios Hernández

Introducción

Los inicios de la supercomputación se remontan a la Segunda Guerra Mundial con proyectos de carácter militar como ENIAC (1943), de física nuclear como CDC 6600 (1964), de propósito general como el ILLIAC-IV (19966-1976) y Cray-1 (1976) (A very brief history of High-Performance Computing, 2016). Con la llegada del nuevo milenio el afán por acelerar la velocidad de los procesadores se reemplazó por el de aumentar el número de núcleos de procesamiento, lo que condujo a los clústeres híbridos. El desarrollo GPGPU y otros aceleradores de propósito general ubican a este tipo de arquitectura como la más eficiente a la hora de integrar tareas de diferentes niveles de complejidad.

Los sistemas embebidos a diferencia de los ordenadores de uso general, realizan tareas predefinidas que responden a restricciones como tiempo real, portabilidad y rendimiento. El área de investigación en sistemas embebidos tiene gran potencial en cuanto se tiene control sobre el diseño hardware y software y por lo tanto de la optimización de sus recursos. Uno de los últimos campos en los que se ha especializado a los sistemas embebidos es en el área de la medicina a través de la computación en imágenes médicas.

Tradicionalmente en la asistencia médica y el transporte asistencial los equipos biomédicos han realizado este tipo de procesamiento de manera fiable junto a otros equipos, como los de medición, consumibles y de asistencia general. Actualmente, el transporte asistencial medicalizado cuenta con los equipos más complejos y sofisticados del mercado, como electrocardiógrafo, desfibrilador automático externo y capnógrafo, que se encargan del

monitoreo, diagnóstico y tratamiento de actividades como: ritmo cardiaco, paro cardiorrespiratorio, taquicardia ventricular y presión parcial de dióxido de carbono en la respiración. De acuerdo a las normas que estipulan la dotación de las ambulancias básicas, no es vital la presencia de estos equipos, porque con el primer tipo de transporte se suplen todo tipo de necesidades. Contrariamente a lo que se establece, la eficacia de este modelo ajustado a la realidad es insuficiente y se evidencia en varios estudios (Proyecto Estándar, 2015) (Proyectos TIPO, 2016) (Plan de Desarrollo Departamental, Suárez , 2016) que muestran las falencias en el transporte asistencial básico y las consecuencias generadas por la brecha de recursos y la ausencia de equipos tecnológicos complejos respecto al transporte asistencial medicalizado y que no pueden ser reemplazados por un monitor de signos vitales, el único elemento de orden tecnológico en las ambulancias básicas. Por varias propiedades que estos equipos poseen, como tamaño, portabilidad, consumo de energía y finalidad, se pueden asociar a sistemas embebidos, lo que permite simular sus funcionalidades de manera confiable en estos sistemas.

La importancia de probar el rendimiento de aplicaciones médicas y diseñar arquitecturas software que aprovechen las características hardware de supercomputadores para apoyar servicios en el área de transporte asistencial como lo propone este trabajo de investigación, yace en la universalidad de su uso, no sólo para el sector académico o entidades privadas de salud, sino también para beneficio del sector público, donde la disposición de equipos de alto desempeño se limita por su costo. El trabajo a futuro aquí sugerido desafía el sistema actual con la mejora - desde una perspectiva tecnológica – de la calidad de servicios que las ambulancias básicas y los prestadores de este servicio ofrecen a la población colombiana.

A lo largo de siete capítulos se expone la tipología de estos vehículos en diferentes panoramas, se profundiza sobre características de sistemas embebidos y el diseño de la

arquitectura software. Se define la metodología software implementada y se muestra el paso a paso del prototipo construido sobre la plataforma embebida que se perfila como apoyo a la atención paramédica de ambulancias. Finalmente se valida su desempeño a través de aplicaciones en el área de la asistencia médica prehospitalaria y de ahí se parte para proponer posibles trabajos futuros.

1. Objetivos

1.1 Objetivo general

Proponer una plataforma basada en una arquitectura embebida de alto desempeño, para apoyar la atención paramédica en vehículos de transporte asistencial básico que mejore la calidad del servicio prestado.

1.2 Objetivos específicos

- Identificar las necesidades TIC del sistema de transporte asistencial de pacientes en Colombia
- Proponer una arquitectura software extensible que permita construir una plataforma para brindar una solución tecnológica a las necesidades identificadas
- Desarrollar un prototipo funcional de la plataforma que valide la arquitectura propuesta
- Validar la plataforma software a través del despliegue del prototipo funcional en una arquitectura embebida de alto desempeño

2. Estado del arte

En este capítulo se identifican las fronteras del conocimiento propuestas hasta ahora con respecto a la investigación en el área por medio de dos perspectivas: investigación relacionada en artículos científicos y aplicaciones implementadas.

2.1 Investigación relacionada

Los autores en (Janani & Minho, 2009) describen el sistema de autenticación probabilística de 17 individuos a los que se les dispuso un sistema basado en ECG y acelerómetro que puede identificarlos de manera continua bajo diversas condiciones que generen actividad física.

En (Mathew , Shane, & Huseyin, 2009) los autores crean un modelo de simulación de operaciones de EMS para evaluar el desempeño de estos sistemas en eventos discretos que abarcan el ciclo: llamada de emergencia, llegada de la ambulancia a la escena, tratamiento de paramédicos a paciente, transporte al hospital, transferencia del paciente a personal del hospital y retorno de la ambulancia a la base.

En (Yuhong , et al, 2015) los autores ubican las estaciones de ambulancias usando información del tráfico en tiempo real para minimizar el tiempo promedio de viaje para alcanzar las peticiones de emergencias, eso lo logran al estimar el tiempo de viaje de los segmentos por carretera utilizando trayectorias GPS reales y proponiendo un refinamiento eficiente basado en PAM para el problema de localización

En (Cem, Kamil, & Ozgul, 2012)los autores se proponen el monitoreo remoto de pacientes para la mitigación de problemas de diabetes, arritmias e hipertensión a través de la conexión inalámbrica basada en el módulo de comunicación Turkcell M2M (machine-to-machine).

En (Frank , Thomas , et al , 2008) se propone MIPGATE, una puerta de acceso a conectividad móvil para aplicaciones vehiculares en redes heterogéneas que se despliega a través de StrokeNet, para la detección temprana de derrames cerebrales usando audio en tiempo real y videoconferencias.

En (Tan-Hsu , Ching-Su, et al , 2009) los autores crean un monitor ECG portable basado en Linux, con doce derivaciones, conectado a un ELP vía USB donde se obtiene una interfaz para almacenamiento, procesamiento de señales y pantalla. La intención de su uso es personal y permite monitorear las condiciones básicas de salud en cualquier momento.

Los autores en (Brian A. , Ahsan H. , & Jim, 2009) diseñan los electrodos y el amplificador de un monitor ECG de bajo costo para países en desarrollo con el fin de minimizar costos y se utilicen componentes reutilizables. Se diseñó especialmente para diagnósticos cardiacos, frecuencia respiratoria y síndrome de apnea del sueño, información que se obtiene a partir de las señales eléctricas.

Los autores en (Fayyaz A., M. U. , & M. , 2008) incorporan la teoría de lógica difusa al clasificador de vecinos más cercanos para reducir el número de prototipos almacenados para minimizar la memoria y las restricciones de tiempo de cómputo. Lo que permite obtener un proceso de decisión más flexible y adaptable al ruido de los datos.

2.2Aplicaciones

2.2.1 Digisens. (Digisens, 2016). En su módulo Healthcare Imaging, Digisens ofrece soluciones a partir de aceleración GPU como gran velocidad de reconstrucción en 3D, algoritmos iterativos para la reducción de la dosis de radiación en la toma de rayos X, procesamiento de volúmenes de datos muy grandes y resultados en tiempo real (cientos de gigabytes en E/S).

2.2.2 Ambulance to Go (Moreno, 2016). Ambulance To Go es una aplicación Móvil y Web para la prestación de servicios médicos ambulatorios y de emergencia permitiendo atender solicitudes entrantes desde páginas web, Apps o llamadas telefónicas recibidas por el Call Center en Bogotá, Cundinamarca. El Software almacena el historial del paciente, Catálogo de Servicios Médicos, y su personal de apoyo tales como médicos, enfermeras y ambulancias para prestar la atención médica tanto en las instalaciones de la entidad médica como en cualquier lugar de la geografía nacional. También identifica las rutas mediante geolocalización y lo cual permite que el personal médico reciba las indicaciones de las rutas más rápidas a través de la aplicación móvil.

2.2.3 Widetech Colombia. (Arbeláez, 2008). Plataforma virtual que realiza el monitoreo y asignación de ambulancias registradas en el Sistema Integral de Comunicación de Ambulancias (SICO) en Cali, Valle del Cauca. El sistema de georreferenciación detecta la ambulancia más cercana al lugar de la emergencia y posteriormente se genera el despacho. La plataforma también permite el control del registro de IPS, para permitir la distribución del equipo de atención de acuerdo con las necesidades requeridas por el usuario.

2.2.4 Alembox (Abdul-Malek, 2014). AlemHealth es un proveedor de servicios de telemedicina de bajo costo y de alta calidad, que conecta hospitales principalmente en Kabul, Afghanistan (y a mercados emergentes) con una red global de radiólogos.

Alembox, es una plataforma de alto desempeño creada por AlemHealth, basado en la NVIDIA Jetson TK1, a donde llegan imágenes de tomografía computarizada, rayos X y resonancia magnética tomadas en los centros médicos locales para ser procesadas por el chip Tegra K1, después éstas son enviadas a un especialista en Estados Unidos, Europa o India a través de una conexión móvil que usa protocolos de bajo ancho de banda y finalmente la evaluación de las mismas es reenvíada en menos de 90 minutos. AlemBox dentro de otras funcionalidades incluye GPS y conectividad 3G y algoritmos de machine learning en los cuales se entrenan datasets que se contruyen a partir de las imágenes, historias clínicas y planes de tratamiento.

Figura 1 AlemBox de AlemHealth.

2.2.5 TechniScan WBU (Whole Breast Ultrasound System) (TechniScan , 2016). Escáner que optimiza tiempos de respuesta, procesamiento y diagnóstico de mamografías través de dos GPUs NVIDIA Tesla D870, cuyo sistema de dispersión inversa FORTRAN y MPI permite la

ejecución del algoritmo en menos de media hora, lo que anteriormente tardaba hasta una hora en un cluster de dieciséis ordenadores con procesadores INTEL Xeon.

Figura 2 TechniScan.

3. Marco teórico

La propuesta que se presenta en este trabajo de investigación busca atacar el problema a través de la unión de tres puntos: Sistemas embebidos, Arquitectura Software y Dotación de Ambulancias, a fin de precisar el ámbito de acción se presentan las siguientes definiciones.

3.1 Sistemas embebidos

3.1.1 Definición Los sistemas embebidos son computadores que se han construido de manera integrada para resolver sólo algunos problemas muy específicos, a menudo no tiene dispositivos perifericos de entrada y salida, por lo que su comunicación se realiza a través de interfaces estándar de cable o inalámbricas. Los sistemas embebidos son sistemas electrónicos e

informáticos, autónomos, que se utilizan cada vez más para contorlar sistemas complejos que se encuentran en la industria automotriz, aeronáutica, aeroespacial, en la industria de automatización, telecomunicaciones, hogares inteligentes, equipamiento médico y de salud (N. & B., 2016).

Un tipo muy frecuente de tecnología que se encuentran en los sistemas embebidos son los sistemas en chip (System-on-a-Chip) o SoC, donde se integran todos o gran parte de los módulos que ponen un sistema informático o electrónico en in circuito integrado o chip (CPU+GPU+ISP).

3.1.2 Características (Jain, 2016) Son específicamente diseñados para realizar una función.

- Óptimos en consumo de energía, tamaño de código, tiempo de ejecución, peso, dimensiones y costo, pues son pensados para ser producidos en masa.
- Se diseñan comúnmente para resolver restricciones en tiempo real
- A menudo interactúan con el mundo externo a través de sensores y agentes, por esto se ejecuta a un ritmo que está determinado por el entorno en el que se desenvuelve.
- Generalmente tienen una interfaz de usuario reducida.
- Están limitados en funcionalidades hardware (memoria de almacenamiento, disco duro) y software (menor número de aplicaciones, aplicaciones reducidas, dejar de lado el sistema operativo o tenerlo muy limitado, menor abstracción a nivel de código) en relación con un computador de escritorio (PC).
- Flexibles a ajustes, una vez se identifique la modificación que se desea realizar, resulta mucho más sencillo modificar una línea de código al software del sistema embebido que reemplazar todo el circuito integrado.

3.1.3 Hardware embebido Uno de los componentes más importantes antes de proponer y evaluar una plataforma software es identificar la arquitectura hardware sobre la que se va a ejecutar, dado que el hardware provee los recursos fundamentales de un sistema como su velocidad de procesamiento, conectividad, capacidad en memoria, rendimiento y eficiencia energética, así como también compatibilidad y fiabilidad según su propósito de diseño.

Ampliando este panorama y con el fin de dar soluciones óptimas y eficaces a necesidades particulares de la vida cotidiana, como el aprovechamiento de los criterios anteriormente mencionados del sistema, se crearon los sistemas embebidos, computadores especializados que cumplen funciones predefinidas generalmente con requisitos específicos; sus aplicaciones desde lo más más general y asequible como los juegos de video y entornos gráficos en la industria automotriz, hasta lo más complejo y costoso como controladores de vuelo, sistemas de guías de misiles, y en el sector de la salud con la robótica quirúrgica y los equipos médicos. El sistema embebido contiene exclusivamente los módulos necesarios para su funcionamiento, de modo que aporta valor agregado a un producto, y proporciona una solución precisa, rápida, y de costo reducido.

3.1.4 Exponentes Las casas fabricantes de estas plataformas son numerosas y cada año más competitivas. Las dos líneas que se trabajan en el grupo de investigación se detallan a continuación.

INTEL propone varias plataformas de desarrollo, algunas de estas son la tarjeta Intel/Galileo Generación 1 y 2 (GEN 1, GEN 2) cuyo propósito es la creación de prototipos, Intel/Edison por su parte, fue diseñada para productos, Intel/Joule es la última SoM para desarrollo IoT.

Tabla 1 *Sistemas embebidos familia INTEL*

Intel	GALILEO GEN 2	EDISON	ARDUINO 101	JOULE 570X
GPU	--	--	--	Intel HD GPU
CPU	Procesador Intel Quark 32 bits	Procesador Intel Atom Dual-Core 32 bits	Procesador Intel Curie 32 bits ARC	Procesador Intel Atom Quad-Core de 64 bits
Velocidad	400 MHz	500 MHz	32 MHz	1.7 GHz
Memoria	512 KB SRAM	1 GB	24 KB SRAM	4 GB
Flash	8 MB	4 GB	196 KB	16 GB eMMC 5.0
Conectividad	Puerto mini PCIe, USB 2.0, micro USB 2.0.	Wi-Fi, Bluetooth	Bluetooth LE	Wi-Fi, Bluetooth.
Almacenamiento	MicroSD 32 GB, USB	4 GB eMMC, SD	Flash 8 MB, MicroSD 8 KB	16 GB eMMC
Sistema operativo	Yocto Linux 1.4, VxWorks, Windows IoT	Yocto Linux 1.6, Poky	RTOS	Linux Kernel 4.4 IoT, Ubuntu 16.04 (beta), Windows 10 IoT
Consumo energético	7-15 V CC	7-15 V CC	3.3 V CC	12 V CC
Precio	$79.90	$50	$30	$369
Año	Agosto 2014	Septiembre 2014	Octubre 2015	Agosto 2016

Nota: Adaptado de (INTEL , s.f.), (Digi-Key Corporation , s.f.), (INTEL, 2017), (INTEL, 2014), (Bhuy L. (INTEL), 2016), (Arduino, s.f.)

NVIDIA con sus últimas tres entregas Jetson TK1, TX1, TX2 a la industria de la supercomputación busca incrementar performance con procesadores más eficientes y mejorar una limitante presente a lo largo del tiempo: memoria RAM.

Las tres tarjetas han aplicado en áreas como la robótica, la visualización de gráficos en 3D de alta calidad, las imágenes médicas, automóviles auto dirigidos, drones, reconocimiento de objetos, entre muchas más.

Tabla 2 *Sistemas embebidos familia NVIDIA*

NVIDIA	JETSON TK1	JETSON TX1	JETSON TX2
GPU	Nvidia Kepler 192 CUDA cores	Nvidia Maxwell 256 CUDA cores	Nvidia Pascal 256 CUDA cores
CPU	Procesador 4-Plus-1 Quad-Core 32 bits ARM Cortex-A15	Procesador Quad-Core 64 bits ARM A57	Procesador Dual-Core NVIDIA Denver2.0 + Quad-Core ARM A57
Velocidad	2.3 GHz	72 MHz	1.3 GHz
Memoria	2 GB bus de 64 bits	4 GB	8 GB
Almacenamiento	16 GB NAND Flash (SD, SATA)	16 GB eMMC	32 GB eMMC
Conectividad	Puerto Gigabit Ethernet, miniPCIe, USB, microUSB	Wi-Fi, Bluetooth 4.1, Gigabit Ethernet, USB 3.0+ USB 2.0	Wi-Fi, Bluetooth 4.1, Gigabit Ethernet, USB 3.0 + USB 2.0
Sistema operativo	Linux4Tegra, Ubuntu Linux 14.04	Linux4Tegra, Ubuntu Linux 14.04	Linux4Tegra, Ubuntu Linux, ROS
Consumo energético	12 V CC	7.5 W	7.5 W
Precio	$198	$299	$399
Año	Abril 2014	Noviembre 2015	Marzo 2017

Nota: Adaptado de (Bongi, 2017), (NVIDIA , 2014), (NVIDIA , 2014), (NVIDIA, 2014)

3.1.5 NVIDIA Jetson TK1 En busca de impulsar el desarrollo de una nueva generación de aplicaciones que requieran visualización y procesamiento de datos en tiempo real, se crea la plataforma móvil de desarrollo para sistemas embebidos, NVIDIA Jetson Tegra K1. Su potencia a nivel de procesamiento con 192 núcleos completamente programables incorporados en la arquitectura de GPU Kepler con un rendimiento de 326 gigaFLOPS y varias opciones de conectividad dentro de los que incluye puertos USB, HDMI, Gigabit Ethernet, puertos SATA y extensión para tarjetas SD; con estas características se posiciona como el primer supercomputador para sistemas embebidos y se elige para el desarrollo de aplicaciones de alta calidad visual contra un nivel considerable de confiabilidad.

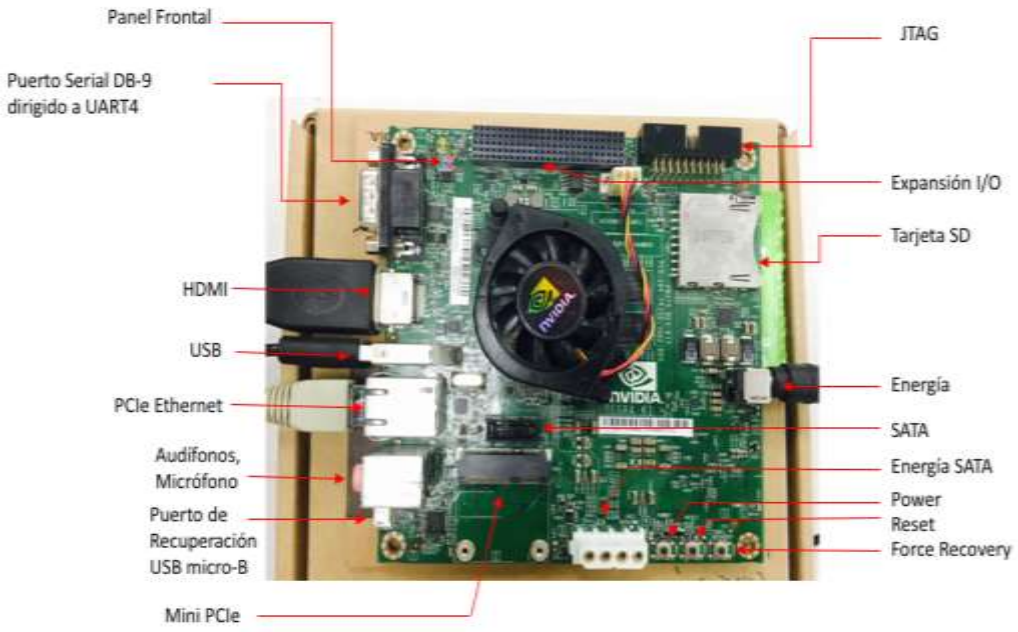

Figura 3 Tarjeta NVIDIA Jetson TK1

Tabla 3 *Especificaciones NVIDIA Jetson TK1*

Componente	***Característica***
GPU	NVIDIA ® Kepler™ 192 CUDA Cores
CPU	NVIDIA 4-Plus-1™ Quad-Core 32 bits ARM Cortex-A15 (Velocidad máxima de reloj de 2.3 GHz)
DRAM	2 GB en bus de 64 bits
ROM	4 MB
Almacenamiento	16 GB NAND Flash (SD, SATA)
Conectividad	Realtek 10/100/1000 Gigabit Ethernet
Imagen	LCD (3840x2160), HDMI (4k: UltraHD 4096x2160)
APIs	OpenGL 4.4, OpenGL ES 3.1, OpenCV4Tegra
Consumo Energético	De 9.5 a 12 Voltios (Corriente Continua)

Nota: Adaptado de (NVIDIA , 2014)

3.2 Arquitectura software

3.2.1 Definición (Somerville, 2010) La arquitectura software de un programa o un sistema computacional es la estructura (o las estructuras del sistema) que comprende los elementos software, las propiedades externamente visibles de estos elementos y la relación entre ellos. Se conoce como "propiedades externamente visibles" a los supuestos que un elemento hace sobre otro, como los servicios que este provee, características de desempeño, manejo de errores, uso de los recursos compartidos, etc. De manera general, la arquitectura es la estructura de los componentes del sistema, sus interrelaciones y los principios y directrices que regulan su diseño y evolución en el tiempo.

3.2.2 Características

- La arquitectura funciona como una herramienta de comunicación, razonamiento, análisis y crecimiento para los sistemas
- Es principalmente una abstracción de un sistema que omite detalles de los elementos que no afectan el uso, cómo son usados y cómo se relacionan con otros elementos.
- Una arquitectura software es desarrollada como primer paso hacia el diseño de un sistema que tiene una colección de propiedades deseadas.
- No hay arquitectura buena o mala, las arquitecturas software son más o menos aptas para algún propósito establecido.
- La arquitectura software se puede evaluar, pero sólo dentro del contexto de metas específicas.

- La arquitectura es más que el resultado de los requisitos funcionales del sistema.

3.2.3 Reglas para el diseño Para el diseño de una arquitectura software se deberían tener en cuenta una serie de parámetros.

Tabla 4 *Reglas para el diseño arquitectura software*

RECOMENDACIONES DE PROCESOS	RECOMENDACIONES DE PRODUCCIÓN O ESTRUCTURALES
La arquitectura debe ser el producto de un solo arquitecto o pequeño grupo de arquitectos con un líder.	La arquitectura debe incluir módulos bien definidos cuyas responsabilidades funcionales se asignan en los principios de ocultamiento de información y la separación de preocupaciones.
El arquitecto debe tener los requisitos funcionales del sistema y una lista articulada de atributos de calidad que la arquitectura debería satisfacer	Cada módulo debe tener una interfaz bien definida que encapsule o esconda los aspectos editables de otro software que use estas facilidades
La arquitectura debe estar bien documentada con al menos una vista estática y una dinámica	Los atributos de calidad deben lograrse usando tácticas específicas arquitectónicas bien conocidas para cada atributo
La arquitectura debe ser distribuida a los actores del sistema quienes deberán estar activamente comprometidos en su revisión.	La arquitectura nunca dependerá de una versión comercial de un producto o herramienta. Si depende de un producto comercial en particular debe ser tan estructurada que al cambiar a otro producto sea sencillo y económico.
La arquitectura debe ser analizada para medidas cuantitativas aplicables y formalmente evaluada para atributos de	Los módulos que muestran datos deben separarse de los módulos que consumen datos, lo cual incrementa su flexibilidad de

RECOMENDACIONES DE PROCESOS	RECOMENDACIONES DE PRODUCCIÓN O ESTRUCTURALES
calidad antes que sea muy tarde cambiarla	ajuste.
La arquitectura debe prestarse a la aplicación gradual a través de la creación del 'sistema óseo', en el que se realicen las vías de comunicación, pero que al principio tiene de funcionalidad mínima.	Para el sistema de procesamiento en paralelo, la arquitectura debe ofrecer procesos o tareas bien definidas que no necesariamente reflejan la estructura del módulo de descomposición. Es decir, los procesos pueden entrelazarse con más de un módulo y un módulo puede incluir procesos que se invocan como parte de más de un proceso.
La arquitectura debe dar lugar a un conjunto específico de las áreas de contención de recursos, cuya resolución se especifica, distribuye y mantiene claramente.	La arquitectura debe incluir un pequeño número de patrones de interacción. El sistema debe hacer las mismas cosas de la misma manera en todas partes. Esto ayudará a la comprensibilidad.

Nota: Reglas para el diseño de Arquitectura Software. Adaptado de (Somerville, 2010)

3.3 Dotación ambulancias

Siendo la dotación de equipamiento biomédico de los vehículos de transporte asistencial el eje central sobre el cual gira el desarrollo de los objetivos planteados, es necesario entrar a mirar detalladamente el estado actual de los vehículos prestadores de servicios asistenciales a nivel nacional para contrastarlo con el panorama a nivel mundial de los países líderes en este campo.

3.3.1 Unión Europea (World, 2015). La Unión Europea a través del Comité Europeo de Estandarización (Comité Européen de Normalisation) publica el estándar CEN 1789:2007- Medical Vehicles and their Equipment- Road Ambulances, donde se unifica en todos los países miembros (Alemania, Austria, Bélgica, Bulgaria, Chipre, República Checa, Croacia, Dinamarca, Eslovequia, Eslovenia, España, Estonia, Finlandia, Francia, Grecia, Hungría, Irlanda, Italia, Letonia, Lituania, Reino Unido*[1], Rumania y Suecia) el diseño, requerimientos, pruebas y equipamiento para ambulancias terrestres; su clasificación de acuerdo a su nivel de emergencia es la siguiente:

En el caso internacional el sistema de servicios médicos de emergencia para países que hacen parte de la Unión Europea como España, se rige de acuerdo a la clasificación del tipo de servicio que se presta. La clasificación se estructura bajo dos pilares fundamentales: el primero, el servicio no asistencial, cuya distribución cobija a la clase A, que a su vez comprende las clases A1 y A2, la primera establece el sistema de transporte convencional donde se facilita el transporte de un paciente que no está en emergencia, por otra parte, en la segunda categoría se adecua el transporte colectivo, donde se asiste pacientes cuyo traslado no es de carácter urgente. El segundo pilar ampara el servicio asistencial, cuya distribución relaciona dos bifurcaciones: la primera es la clase B donde se proporciona soporte vital básico y atención sanitaria inicial, para dicho fin se emplea una ambulancia de emergencia; la clase C, que proporciona soporte vital avanzado y monitoreo. Para estas dos últimas clases en cualquiera que sea el caso, los vehículos están dotados con dispositivos de transmisión de datos, localización GPS con su centro de urgencias y equipamiento médico básico para vehículos de emergencia.

[1] Reino Unido hace parte de la Unión Europea hasta junio de 2016, cuando se llevó a referendo (Brexit) su permanencia con la pregunta: "¿Should the United Kingdom remain a member of the European Union or leave the European Union? "donde el 51,9% de los votantes eligió la opción de retiro.

Entre tanto, países como Alemania, Portugal, Reino Unido, Grecia y Lituania están siguiendo el proyecto de "Servicios Médicos de Emergencia a Nivel Mundial" el cual se divide en dos modelos principales: el franco-alemán, en el que una ambulancia es atendida por los médicos, y el modelo anglo-americano, donde las ambulancias cuentan con personal de técnicos sanitarios entrenados en soporte vital básico (BLS), soporte vital intermedio (ILS) o soporte vital avanzado (ALS). En ambos modelos, busca utilizar técnicas modernas, herramientas y tecnologías para reducir las incertidumbres médicas, con el fin de alcanzar la mayor calidad posible de atención prehospitalaria centrada en el paciente.

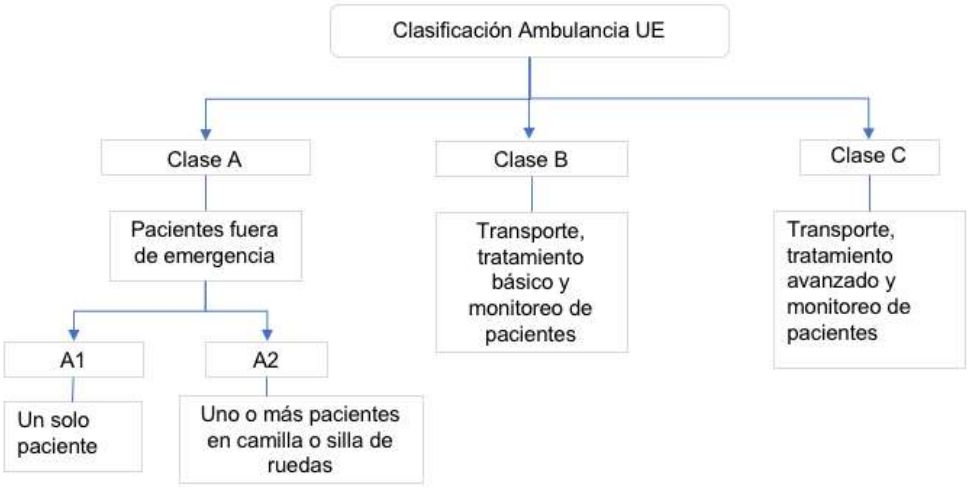

Figura 4 Tipología ambulancias Unión Europea

3.3.2 Estados Unidos (National EMS Assessment, 2011, p. 63) (Federal Specification for the Star-of-Life Ambulance. U.S. , 2007). Los sistemas de servicios médicos de emergencia (EMS) se configuran de manera diferente dependiendo de varios factores que incluyen: tamaño, demografía, geografía y la política de los Estados y comunidades locales a las que sirven. Aunque existe información sobre la organización, financiamiento y entrega de EMS en 200 de

las ciudades más grandes de la nación, esta información es incompleta y no proporciona ninguna información sobre cómo se organizan los servicios para la proporción de la población de la nación (75%) que reside fuera de estas áreas urbanas.

En todo caso, existen entidades que estandarizan el servicio de ambulancias básicas (BLS) y avanzadas (ALS). La Administración de Servicios Generales (General Services Association, 2010) y La Asociación Nacional de Protección contra el Fuego (NFPA 1917, 2015) son las encargadas de estandarizar la dotación y estandarización de este tipo de vehículos al categorizarlos por GVWR (Gross Vehicle Weight) o capacidad máxima de peso operativo.

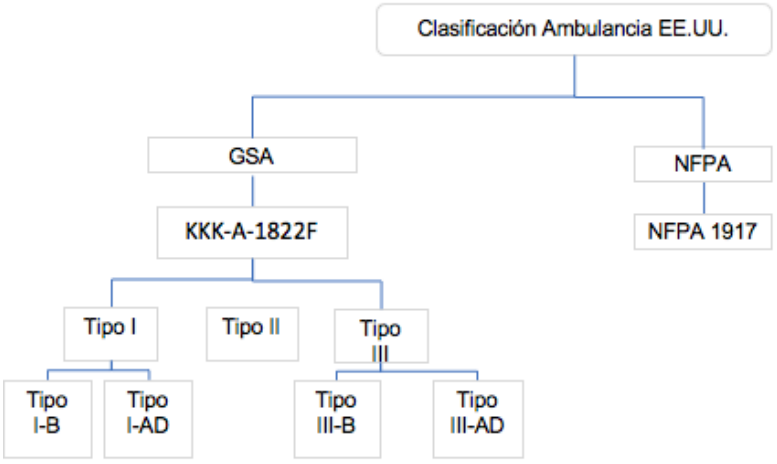

Figura 5 Tipología ambulancias Estados Unidos

3.3.3 Colombia La provisión de servicios de salud se efectúa con la prestación de servicios básicos y complejos a la población por parte del Sistema General de Seguridad Social en Salud, que desde el Ministerio de Salud y Protección Social coordina las Secretarías de Salud territoriales, en las cuales, de manera autónoma se diseña e instaura la Red de Urgencias, Emergencias y Desastres y la estructura de la Atención Prehospitalaria.

En las Entidades Promotoras de Salud se gestiona la afiliación, procesos administrativos y comerciales, mientras que las Instituciones Prestadores de Servicios de Salud se hacen cargo de los servicios complejos. El traslado de un paciente en un medio de transporte terrestre, aéreo, marítimo o fluvial es lo que se conoce como Transporte Asistencial, y busca dar atención oportuna y adecuada a la persona en cuestión durante el desplazamiento a través de dotación básica.

En el caso del transporte terrestre, conocer la dotación de la ambulancia es muy importante porque se enmarca dentro del componente de prestación de servicios de la Política Nacional de Prestación de Servicios del Ministerio de Salud y Protección Social, por lo que la habilitación de los prestadores de este tipo de servicio está directamente relacionada con los equipos de dotación con los que cuenten.

n el orden de menor a mayor complejidad, el primer tipo de transporte en el que operan estos vehículos es el Transporte Asistencial Básico (TAB), aquí la ambulancia es la unidad móvil destinada al traslado de pacientes cuyo estado potencial o real de salud no precisa cuidado asistencial médico durante la atención y transporte, por ello se encuentra bajo responsabilidad de un auxiliar de enfermería o técnico en atención prehospitalaria. En un nivel más complejo se encuentra la ambulancia de Transporte Asistencial Medicalizado (TAM), conocida como la unidad de intervención con equipo avanzado tripulada por un médico entrenado, enfermera, auxiliar o tecnólogo en atención prehospitalaria.

En cuanto a las especificaciones técnicas, estas ambulancias deben tener en cuenta una lista general y específica de equipos estándar, reglamentados tanto en (Ministerio de Salud y Protección Social, 2014) como en la (Norma Técnica Colombiana, 2007, p. 19, p. 20) que definen la tipología vehicular para ambulancias de transporte terrestre, y el (Decreto 1471, 2014)

"Por el cual se reorganiza el Subsistema Nacional de Calidad y se modifica el Decreto 2269 de 1993", así como también el (Ministerio de Salud y Protección Social, 2013) a través del Instituto Nacional de Vigilancia de Medicamentos y Alimentos - INVIMA reglamenta a todos los equipos biomédicos en el territorio nacional.

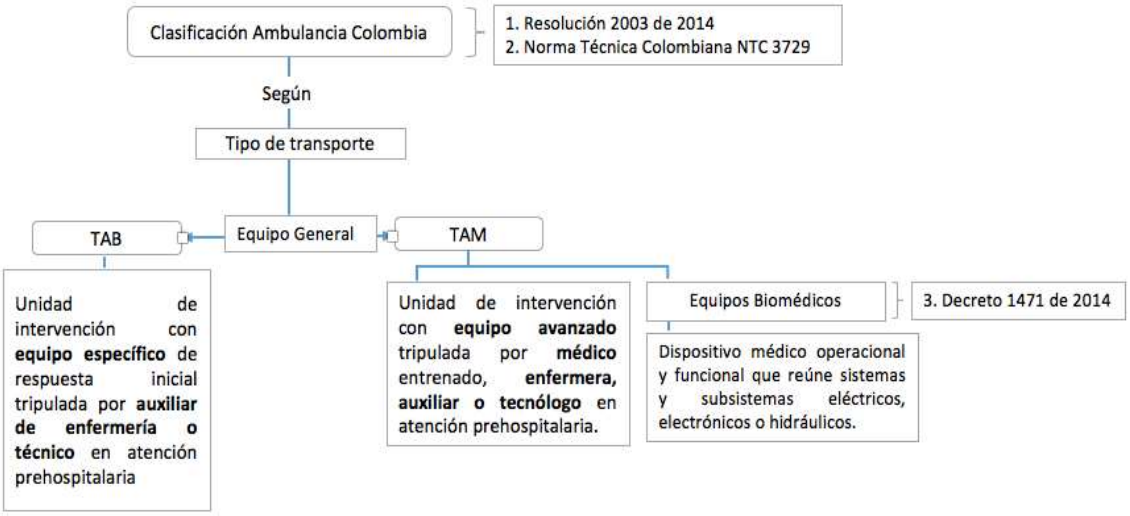

Figura 6 Tipología ambulancias Colombia.

4. Metodología

4.1 Prototipos evolutivos

Un prototipo es un modelo del comportamiento del sistema que puede ser usado para entenderlo completamente o ciertos aspectos de él y así identificar los requerimientos. Aunque no es un sistema completo, posee las características del sistema final o parte de ellas.

Existen varios tipos de prototipos de desarrollo software: prototipado de interfaz de usuario, funcional, modelos de rendimiento, rápido o desechable y evolutivos. Es en éste último donde se hará especial énfasis, al ser el implementado.

4.1.1 Definición Son modelos que se adaptan a la evolución generada por los requisitos del sistema en función del tiempo. Igualmente, se despliegan una serie de versiones sucesivas de un producto (Ingeniería en Sistemas, 2009).

En el desarrollo software existen varias metodologías o paradigmas que estructuran el proceso de construcción de un producto, para el caso particular de este estudio se exponen conceptos básicos de la metodología implementada a lo largo del proceso, una adaptación de prototipos evolutivos, aunque se desarrolla una gran parte software, también se tocan elementos hardware.

4.1.2 Características Se adaptan más fácilmente a los cambios introducidos a lo largo del desarrollo, son iterativos, permiten el aumento de complejidad en cada iteración (evolutivo, 2006).

4.1.3 Ciclo de vida En general, cada modelo de desarrollo software trae consigo un cronograma del cual se desplegarán una serie de actividades a implementar, así como también un ciclo de vida que recorrerá desde el inicio hasta el fin su desarrollo. Particularmente en el caso de la metodología de prototipos evolutivos culmina con la entrega de un producto final, al cual se le conoce como producto de ingeniería. Sin embargo, teniendo en cuenta el caso de estudio que se expondrá a continuación, el ciclo de vida se limita a la entrega de un prototipo funcional validado a través de pruebas funcionales.

Figura 7 Ciclo de vida prototipado evolutivo

4.1.4 Caso de uso Se ha optado por implementar una adaptación de la metodología prototipado evolutivo, pues, aunque el sistema propuesto presenta características propias de un proyecto de desarrollo software, emplea a su vez soluciones hardware y la unificación de estas dos. La metodología se desarrolla en cuatro fases que se presentan a continuación:

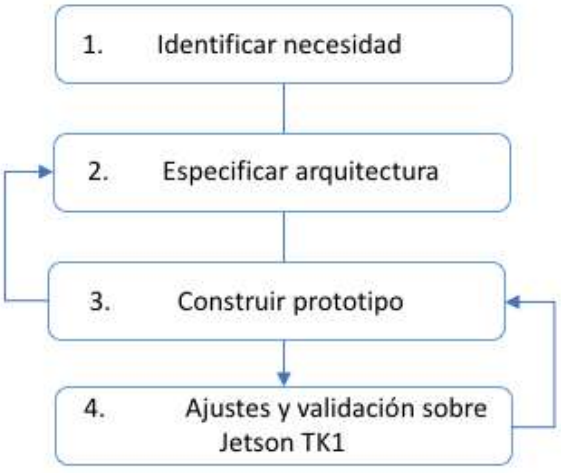

Figura 8 Metodología implementada

Gracias a las características propias del sistema, esta metodología permite a cada fase estar sujeta a modificaciones a lo largo del tiempo de desarrollo del proyecto, lo que posibilita la ejecución de cada fase de forma no lineal.

Fase 1: Identificación de la necesidad. Durante la primera fase se identificaron las necesidades generales y específicas que hay en el entorno. Las actividades que se desarrollaron a lo largo de cada fase incluyen:

- Entrevistas a IPSs relacionadas con el servicio de transporte de emergencias médicas.
- Análisis de la normatividad: Resolución Número 00002003 de 2014
- Análisis de la normatividad: Decreto 1011 de 2006

Fase 2: Especificación de la arquitectura. La segunda fase propone la arquitectura software y sus especificaciones con base en el análisis de las necesidades planteadas durante la fase previa. Dentro del desarrollo de esta fase se contemplan:

- Identificar los atributos de calidad destacados de la plataforma
- Propuesta de diferentes vistas de la arquitectura
- Validación de la arquitectura propuesta

Fase 3: Construcción del prototipo. En la tercera fase se construye el prototipo de la plataforma, durante esta fase se adaptan las especificaciones de la arquitectura constantemente, de acuerdo a las especificaciones de la fase anterior, por lo que a lo largo de esta fase se hace necesario la realimentación de la información. Para el adelanto de esta etapa se llevan a cabo:

- Diseñado detallado de los componentes básicos
- Codificación del prototipo
- Pruebas funcionales de la plataforma

Fase 4: Validación de la plataforma sobre la arquitectura Jetson. En esta última fase se realizan los ajustes y la validación del prototipo sobre la arquitectura embebida Jetson Tegra K1. Para tal fin, se hace necesario:

- Despliegue sobre arquitectura hardware Jetson TK1
- Evaluación prototipo

5. Desarrollo del proyecto

5.1 Necesidades T.I. en ambulancias

5.1.1 Cierre de la brecha tecnológica TAB – TAM En el Traslado Asistencial Básico se cuenta con un monitor de signos vitales con mínimo monitoreo de presión arterial no invasiva, brazaletes adulto y pediátrico, frecuencia cardíaca y oximetría de pulso. En el Traslado Asistencial Medicalizado se dispone de un monitor de transporte multiparámetro de signos vitales que incluya mínimo electrocardiografía, oximetría de pulso, presión no invasiva, temperatura y respiración, un desfibrilador bifásico con capacidad de realizar cardioversión sincrónica y marcapasos transcutáneo (Ministerio de Salud y Protección Social, 2013, p. 17).

5.1.2 Monitor ECG en TAB En la ambulancia asistencial básica se da prioridad a la valoración de los signos vitales y la estabilización de la persona para generar un diagnóstico y prevenir el desarrollo de complicaciones y daños permanentes. Como en el caso del trauma a

nivel prehospitalario, donde los pasos que se deben seguir para dar atención a un dolor torácico dentro de una valoración inmediata (menor a 10 minutos) se encuentran: obtener un electrocardiograma e interpretarlo en cinco minutos, transmitir el trazo al hospital receptor, así como también, una radiografía de tórax portátil (Ministerio de Salud y Protección Social, 2012).

5.1.3 Reducción tiempo de traslado de pacientes En general, menos del 20% de los pacientes con heridas penetrantes del corazón llegan con vida al hospital. Las tasas de supervivencia después de una toracotomía de urgencia se han asociado directamente con el tiempo de transporte hasta el centro asistencial más cercano. Se podrían mejorar las tasas de supervivencia si la toracotomía se realizara en la fase de atención prehospitalaria por personal capacitado, por esto se insiste en que la evaluación de los pacientes debería ser rápida y metódica. (Ministerio de Salud y Protección Social, 2012)

5.1.4 Reducción tasa morbimortalidad en transporte asistencial La principal causa de muerte en los últimos cinco años se atribuye a enfermedades isquémicas del corazón con 32.976, casos certificados equivalentes al 16.3% (Departamento Administrativo Nacional de Estadística, 2014). En 2011 la cifra llegó a 29.000 muertes. Los paros cardiacos repentinos son el resultado de la fibrilación ventricular, lo que significa un ritmo cardiaco acelerado y no sincronizado, que inicia en los ventrículos. En estos casos, el corazón debe ser desfibrilado rápidamente que las posibilidades de sobrevivencia caen entre el 7% y 10% por cada minuto que no se restaure el ritmo cardiaco normal (Ministerio de Salud y Protección Social, 2015) (Piñeros, 2015)

5.1.5 Apoyo al servicio de IPSs Para las instituciones Prestadoras de Servicios es imprescindible mejorar la asistencia remota en: diagnóstico, monitorización, ubicación, vigilancia., disminuir el tiempo de transmisión de información, así como también aumentar el tiempo de vigencia de los equipos tecnológicos (Dos años). (AME 2015)

5.1.6 Optimización de tiempos de traslado y puntos de cobertura de ambulancias Algoritmo de Búsqueda en vecindades (Variable Neighborhood Search, VNS) para localización de ambulancias. Ambulance location and relocation problems with time-dependent travel times. (Varena & Karl F., 2010)

5.1.7 Uso apropiado de ambulancias Uso inapropiado de ambulancias, análisis desde los sectores privado, público y Medicaid. The emergent problem of ambulance misuse. (E. & J. , 1993)

5.1.8 Reducción de costos Incremento en la demanda y necesidad de reducción de altos costos, los objetivos de desempeño entraron en cambio abrupto en 2016 y continuarán en 2017. No hay suficiente paramédicos para cubrir los servicios solicitados. Es probable que continúen cobrando por los servicios de asistencia prehospitalaria a las personas. The biggest challenge facing ambulances services. (Dave & Alison, 2016)

5.2 Especificación de la arquitectura

Todos los sistemas embebidos funcionan gracias a la implementación de una arquitectura software que relaciona perspectivas de elementos de diseño de alto nivel y que de acuerdo a restricciones generados por el entorno da cumplimiento a requerimientos funcionales y no funcionales del sistema. El objetivo de la arquitectura software es obtener una calidad suficiente de un conjunto de propiedades del software a desarrollar como desempeño, robustez, capacidad de distribución y mantenibilidad del mismo.

El diseño de esta arquitectura se contempla como una serie de decisiones estructurales que se proponen según el prototipo planteado, el enfoque que se dará, el modelo que se implantará y la relación de complementariedad de sus componentes. El modelo de referencia que se siguió para la especificación de la arquitectura fue un modelo a capas o modular.

Figura 9 Arquitectura software propuesta

En orden ascendente, el primer nivel de la arquitectura corresponde al Físico o Hardware: SoC Tegra K1 basado en ARM que integra 192 núcleos en GPU, 5 núcleos para CPU y memoria

compartida. El segundo nivel da inicio al Software, con el Sistema Operativo: Linux4Tegra, Ubuntu 14.04. El tercer nivel se constituye a partir de la construcción de compiladores, que serán los intérpretes de los lenguajes propuestos, JVM y LLVM. C/CUDA es el lenguaje predefinido para la placa de desarrollo, por esto no es necesaria la construcción de un compilador, basta con la instalación del paquete de desarrollo. En el cuarto nivel se encuentran los tres lenguajes de medio (C/C++) y alto nivel (Java, Python). Finalmente, la arquitectura propuesta en su quinto nivel o de Aplicaciones, muestra el despliegue de dos programas de cómputo de alto desempeño y un tercero de carácter general.

5.3 Recursos Software

En este apartado se muestran las herramientas necesarias para el desarrollo del prototipo. Dentro de las más relevantes se encuentran los lenguajes de programación, los entornos de desarrollo y paquetes.

Tabla 5 *Recursos software*

Nombre	Versión	Descripción
NVIDIA Cuda Toolkit	6.5 Ubuntu 14.04 ARMv7	Paquete que agrupa e instala las herramientas software necesarias para desarrollar sobre la plataforma embebida NVIDIA Jetson. Incluye: kit de desarrollo JetPack 2.0 para flashear la tarjeta con las últimas imágenes del SO, herramientas de desarrollo host y destino, APIs, middleware, ejemplos de prueba, documentación.
Python	2.7.10	Lenguaje de programación interpretado, de alto nivel, popular por su sintaxis clara y versátil de código abierto. Dentro de sus extensiones más importantes se encuentran

Nombre	Versión	Descripción
		paquetes como NumPy, SciPy y librerías como Pandas, las cuales permiten desarrollar una gran variedad de aplicaciones en el campo de la computación científica
Miniconda	3.18.3	Versión de la plataforma Conda diseñada para discos con espacio limitado. Incluye conda, Python y paquetes como pip, zlib, entre otros. Más de 720 paquetes científicos y sus dependencias se pueden instalar individualmente desde el repositorio Continuum con el comando "conda install", lo que lo hace mucho más ágil.
Numba	0.29.0	Compilador para arrays y funciones numéricas de Python, que acelera aplicaciones con funciones de alto desempeño escritas en Python. Genera código de máquina optimizado a partir del código Python utilizando la infraestructura del compilador LLVM. Genera código nativo para la CPU e integra GPU con software científico de Python por medio de Numpy
LLVM	3.8 Ubuntu 14.04 armhf	Colección de tecnologías modulares y reutilizables y tecnologías de herramientas en cadena. Sus librerías proporcionan un optimizador fuente-destino, junto con soporte de generación de código. Algunas de sus herramientas incluyen los compiladores: Clang, C/C++/Objective-C.
Python	2.7.10	Lenguaje de programación interpretado, de alto nivel, popular por su sintaxis clara y versátil de código abierto. Dentro de sus extensiones más importantes se encuentran paquetes como NumPy, SciPy y librerías como Pandas.
JVM	25.111-b14 mixed mode	Entorno en el que se ejecutan los programas Java. Garantiza portabilidad de las aplicaciones Java. Define un ordenador abstracto y específica las instrucciones

Nombre	Versión	Descripción
	(ARMv7l)	(bytecodes) que este ordenador puede ejecutar.
JAVA	1.8.0_111	Lenguaje de programación compilado e interpretado, de propósito general, alto nivel y orientado a objetos. Los tipos de programas más comunes son los applets y los servlets.
NetBeans IDE	8.0.2	Entorno de desarrollo que utiliza el JDK junto a las librerías para programar clases en Java. Similares a este entorno se encuentran: Eclipse, BlueJay, J-Box, Java Enterprise Edition.
Matplotlib	2.0.0	Librería para generación de gráficas estáticas de Python y NumPy.
Biosppy	0.3.0	Paquete Python para procesamiento de bioseñales (BVP, ECG, EEG, EMG, Respiración).

Nota: Adaptado de (NVIDIA Embedded Computing, 2014) (Numba , 2015) (LLVM, 2011) (Universidad del País Vasco, 2010) (Conda Documentation, 2015) (Chris, 2010)

5.4 Construcción del prototipo

La construcción del prototipo funcional inicia con el diseño de la arquitectura software, generada en la primera fase de desarrollo. A partir del segundo nivel de este modelo, se inician tres subfases de instalación y test. De manera general se muestran las herramientas necesarias y la descripción de sus dependencias en este capítulo, y de manera puntual el procedimiento paso a paso se presenta en el Apéndice A.

5.4.1 Descripción tecnología CUDA Las tarjetas gráficas NVIDIA vienen inicialmente configuradas con L4T (Linux for Tegra), lo que incluye por supuesto el kernel ARM para

equipos embebidos, además contiene paquetes y librerías como OpenACC, CUDA 6.5, OpenGL instalados. A diferencia de cualquier otro lenguaje, para este solo es necesario la descarga del kit de desarrollo.

5.4.2 Descripción tecnología JAVA Instalación del JDK:

Descarga del JDK 8u45 en el sitio web oficial (Oracle, 2017).

- Linux ARM 32 Hard Float ABI
- Arquitectura ARMV7vl de 32 bits (NVIDIA Jetson TK1)
- Configuración de las preferencias del compilador y la máquina virtual
- Paquete de Descarga: Java SE (NetBeans Platform SDK, Java SE, Java FX).

5.4.3 Descripción tecnología Python Instalación de Miniconda:

Descarga desde el sitio web (Miniconda, 2016)

- Linux armv7l Installer
- Versión 3.18.3

Nota: Durante la instalación de Miniconda 3.18.3 para la versión de Python 2.7.10 en la tarjeta NVIDIA Jetson TK1 los paquetes no incluían Numba, por lo que fue necesario realizar su instalación con la herramienta pip.

1. Instalación LLVM (LLVM, 2016). Requisitos:

- GNU Make
- GCC
- Python
- zlib

2. Instalación de Numba. Requisitos:

- llvmlite armhf
- Numpy
- Funcsigs
- Enum34

5.5 Ajustes y Validación

El proceso de validación determina si los requerimientos y el examen final del prototipo construido cumple su uso intencional específico. La primera comprende el análisis de tiempos de ejecución para el producto de matrices 2-D de tres tamaños diferentes. En la segunda, la de tres aplicaciones, dos de ellas enfocadas en brindar solución a necesidades HPC, y una a necesidades genéricas del sistema.

5.5.1 Marco de comparación tecnológico Una matriz es un conjunto de números colocados en líneas horizontales y verticales y dispuestos en forma rectangular. Pero a pesar de su simplicidad, son uno de los objetos más útiles y fundamentales en computación científica. Las aplicaciones que se realizan a partir de matrices incluyen gráficos computarizados, solución de sistemas de ecuaciones, comparación de secuencia genética, modelado de circuitos eléctricos o redes informáticas, entre otros. Al ser objetos matemáticos, pueden ser sumadas, restadas, multiplicadas y algunas veces divididas. Las aplicaciones finales que se ponen en marcha tienen mucha afinidad con operaciones de productos matriciales, por esto, se realiza una prueba de rendimiento a través **de este cálculo.**

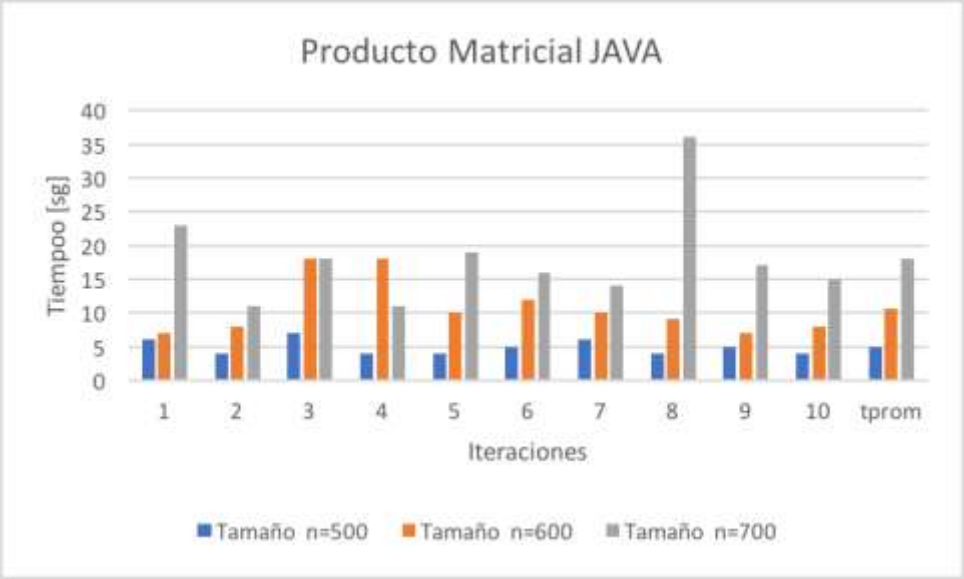

Figura 10 Producto matricial JAVA. Ej.: Multimatrix.java

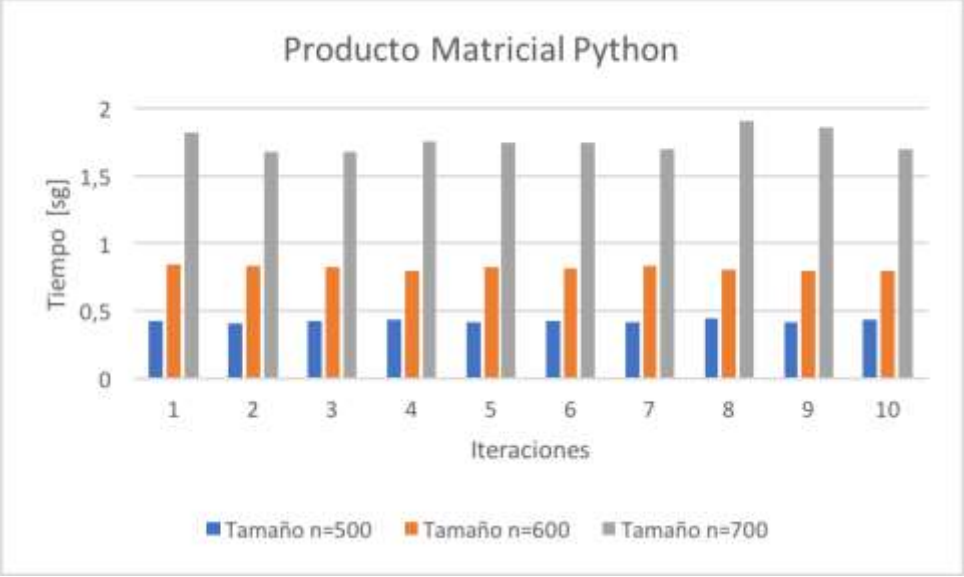

Figura 11 Producto matricial Python. Ej.: matrix.py

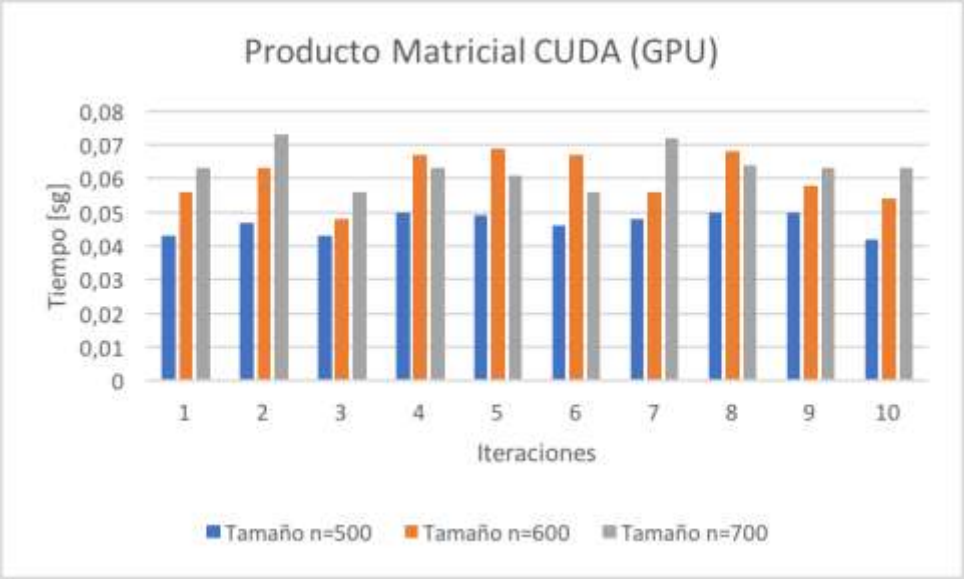

Figura 12 Producto matricial CUDA. Ej.: multmatex2.cu

Análisis de resultados. El análisis del rendimiento de los cálculos en cada lenguaje es posible si se tiene en cuenta el modelo de la arquitectura diseñada, porque cada operación ejecutada es el resultado del uso óptimo o ineficiente de varios componentes hardware.

La primera gráfica muestra el comportamiento del lenguaje Java. La JVM es un programa nativo, es decir, que se ejecuta en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en código binario al que se le conoce como bytecode, este bytecode es generado por el compilador del lenguaje Java: javac. Por esto, el tiempo que tarda en efectuar las multiplicaciones es mucho mayor respecto a los otros dos lenguajes propuestos, al sufrir un retardo entre su interpretación y compilación. Si bien en promedio se trata de tiempos de ejecución dentro de [4.9,18 sg], en la iteración No.8 se alcanzan los 36 segundos para el producto de matrices de tamaño 700^2, valor muy superior frente a otros lenguajes, pero continua favorable para el tamaño de la matriz resultante 490.000.

Numba por su parte, es un compilador basado en LLVM. Está pensado para optimizar códigos numéricos en Python y su extensión Numpy, importada en el código de producto de matrices es fundamental para este proceso. De ahí, que al observar el comportamiento de los tiempos de ejecución respecto al lenguaje CUDA es el más cercano y hasta ahora constante en un intervalo de tiempo [0.4, 1.9 sg] para todas las iteraciones.

Finalmente, como se plantea en la hipótesis de este estudio, CUDA representa un cálculo optimizado de las multiplicaciones de matrices en cada iteración, porque a diferencia de los otros lenguajes el incremento del tamaño de las matrices no alcanza el tiempo de ejecución de 0.1 segundos. Esto se debe al nivel de paralelismo del código en la declaración del kernel y en varios ciclos iterativos, que de no ser ejecutados en GPU representarían un mayor costo computacional para el programa.

5.5.2 Aplicaciones ejemplo.

1.Reducción de ruido en imagen para CUDA en arquitectura ARMv7l

La imagen digital se ha convertido en algo imprescindible en la sociedad actual. Una sola imagen en alta definición puede tener más de dos millones de pixeles. Muchos algoritmos de procesamiento de imágenes requieren decenas de cálculos en punto flotante por pixel, lo que daría como resultado tiempos de ejecución muy altos. La razón principal por la cual se elige la GPU de NVIDIA es porque se puede configurar el kernel para optimizar la asignación de valores de cada pixel de la imagen. La reducción de ruido en imágenes electrocardiográficas desde este lenguaje se valida en la plataforma a través de la reducción de ruido en una imagen convencional en tiempo real.

Descripción. El primer método que se implementó para la reducción de ruido blanco en una imagen fue el filtro KNN (K Nearest Neighbors Filter), donde la idea principal es calcular los pesos de los pixeles dependiendo de qué tan similares son sus colores a través de la ecuación:

$$KNN_{h,r}u(x) = \frac{1}{C(x)} \int_{\Omega(x)} u(y) * e^{\frac{|y-x|^2}{r^2}} e^{\frac{|u(y)-u(x)|^2}{h^2}} dy$$

Donde:

$u(q)$ representa el ruido original de la imagen

$KNN_{h,r}\ u(x)$ es el resultado producto del filtro KNN con parámetros h (grado de filtrado), r (coeficiente de ruido gaussiano).

$C(x)$ es el coeficiente normalización

$\Omega_{(p)}$ es el tamaño del vecindario que rodea a un pixel p (7x7 para este caso)

Los bloques son de tamaño NxN, donde $N = 2M + 1$

El segundo filtro que se implementó fue el de NLM (Non Local Means Filter), que es una variación un poco más compleja del filtro KNN. Para ello, se presenta la siguiente ecuación:

$$NLM_{h,r,B}u(x) = \frac{1}{C(x)} \int_{\Omega(x)} u(y) * e^{\frac{|y-x|^2}{r^2}} e^{\frac{ColorDistance\ (B(x),B(y))}{h^2}} dy$$

Donde:

Los bloques de los pixeles serán de tamaño KxK donde $K = 2L + 1$

$B(q)$ es el vecindario que rodea a un pixel q

$C(x)$, $\Omega(x)$ y $u(x)$ son los mismos valores que en KNN

$ColorDistance(B(x), B(y))$ representa la suma normalizada de las diferencias absolutas entre bloques alrededor del pixel $u(x)$ y alrededor del pixel $u(y)$

$$ColorDistance\big(B(x), B(y)\big) = \frac{1}{S(B)} \int_{B(x)} \left|u\big(y + (x - \alpha)\big) - u(\alpha)^2\right| \, d\alpha$$

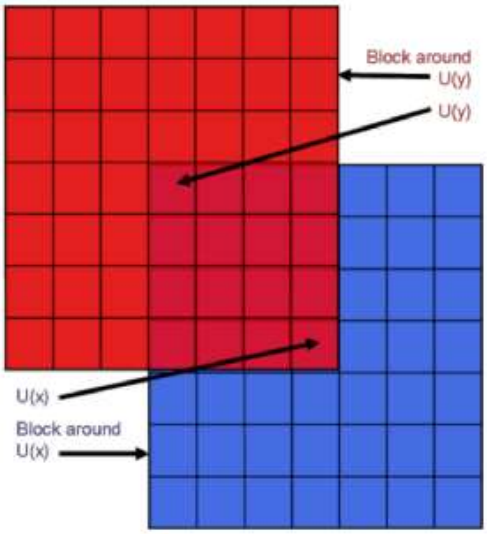

Figura 13 Vecindario alrededor de pixeles $U_{(x)}$ y $U_{(y)}$

Implementación. A continuación, se muestra una imagen con ruido a la cual se le aplicaron los filtros de eliminación de ruido KNN y NLM respectivamente.

Figura 14 Imagen restaurada filtro KNN

Figura 15 Imagen restaurada filtro NLM

Resultados

Tabla 6. *Resultados CUDA*

FPS	LerpC	Ruido KNN	Ruido NLM
4362.80	0.20	0.32	1.45

Donde:

- FPS: Frames Per Second o imágenes por segundo, corresponde a la velocidad o tasa a la cual la Jetson TK1 muestra las imágenes, cuadros o fotogramas.
- LerpC: Lerp Coefficient, corresponde al coeficiente de interpolación lineal o de mezcla entre los pixeles originales y los pixeles restaurados. Sus valores deben estar dentro de (0.00f – 0.33f)

- Ruido KNN: reducción de ruido blanco en imagen de 320 x 408 a 226.1 fps y $\Omega(x)$ = 7x7
- Ruido NLM: reducción de ruido blanco en imagen de 320 x 408 a 10.1 fps y $\Omega(x)$ = 7x7

2.Detección de arritmias en señales electrocardiográficas para Python en arquitectura ARMv7l.

La gran acogida que han tenido los algoritmos de análisis y procesamiento de imágenes cardiológicas y radiológicas se debe al diagnóstico, pronóstico o tratamiento de enfermedades. Las señales ECG (electrocardiográficas) están dadas por la forma de la onda electromagnética PQRSTU. El electrocardiograma es la representación gráfica de la actividad eléctrica del corazón, que se obtiene con un electrocardiógrafo en forma de cinta continua, por su portabilidad se disponen en casi cualquier lugar con fuente de energía. Probar la compatibilidad de la plataforma con Python, un lenguaje de alto nivel a través de la detección de arritmias en señales ECG es posible a través del paquete BioSPPY, el cual adicionalmente al módulo ECG incluye módulos para análisis de bioseñales como BVP, EEG, EMG y Respiración. El correcto funcionamiento de la función ecg.py y el script con los parámetros heartbeats.py se valida la futura detección de arritmias en la plataforma

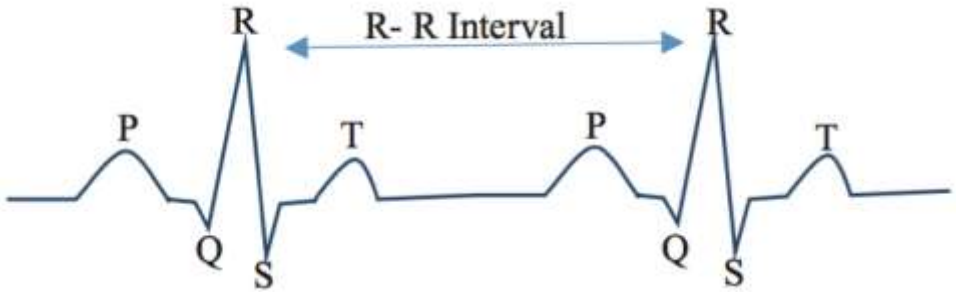

Figura 16 Señal ECG

Donde:

La onda P representa la propagación del impulso eléctrico desde el nodo sinoauricular (SA) a través de las aurículas.

El intervalo PR representa el tiempo necesario para que el impulso se propague sobre la aurícula y a través del nodo auriculoventricular (AV), donde el impulso se detiene por un período corto de tiempo.

El intervalo RR es la distancia entre dos ondas R sucesivas. En el ritmo sinusal este intervalo debe ser constante.

El complejo QRS representa la propagación del impulso a través de los ventrículos (despolarización ventricular).

El segmento ST representa un período de inactividad entre la despolarización y la repolarización ventricular.

La onda T indica repolarización ventricular (La Web del Electrocardiograma, 2013)

Paquete BioSPPY. Kit de desarrollo de procesamiento de bioseñales para Python, agrupa varios métodos de procesamiento de señales y reconocimiento de patrones orientado al análisis de bioseñales. (BioSPPy Documentation, 2015).

Dependencias:

- Bidict
- H5py
- Matplotlib
- Numpy
- Scikit-learn

- Scipy
- Shortuuid

Nota: Todas las dependencias se instalan a través de conda install o pip install, con la excepción de Matplotlib, para la arquitectura armv7l es necesaria la descarga del código fuente en (Python Software Foundation, 2016), opción matplotlib 2.0.0.tar.gz, para posteriormente instalar con: pip install [ruta donde se encuentra descargado el código fuente].

Implementación. A continuación, se muestra en funcionamiento el script heartbeats.py, en el que se llama a la función ecg.py (biosspy.signals, 2015) del paquete Biosppy con 14.999 pulsos y una frecuencia muestral de 100 Hz (Carreiras, 2017):

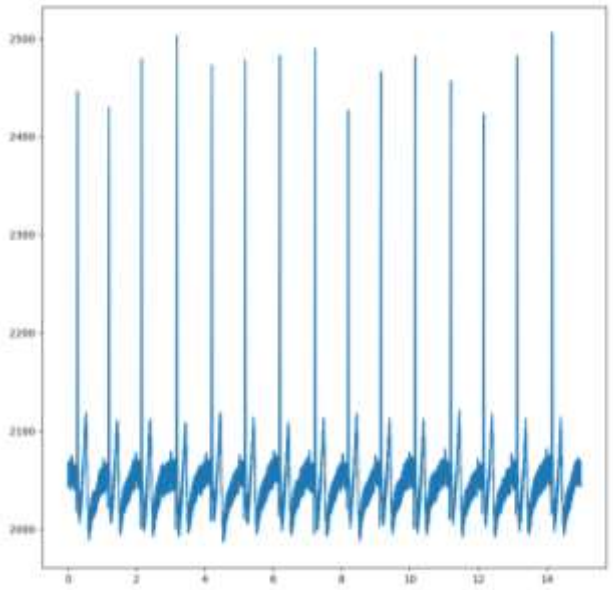

Figura 17 Señal ECG con ruido

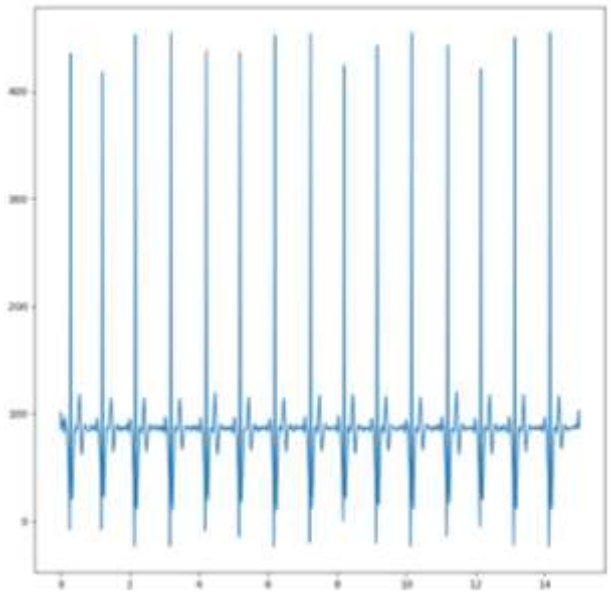

Figura 18 Señal restaurada filtro FIR

Donde la primera figura describe la señal en bruto con presencia de ruido, y la segunda describe la señal procesada con el filtro FIR (Finite Impulse Response). El eje horizontal mide el tiempo que transcurre durante los impulsos cardiacos y el eje vertical el voltaje o amplitud, usualmente en mili voltios.

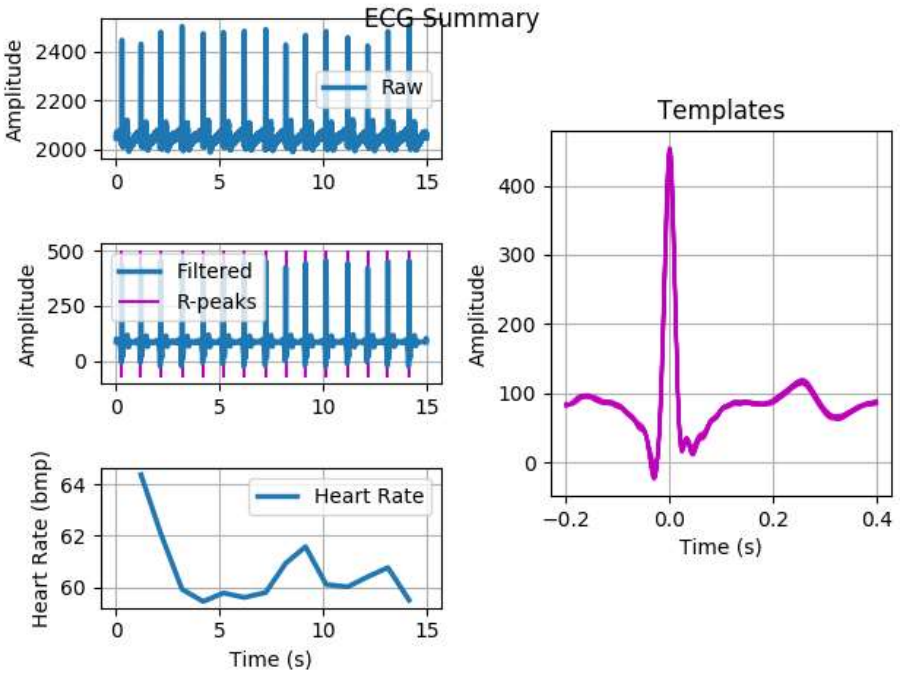

Figura 19 Reporte ECG

La primera figura en la esquina superior izquierda describe la señal inicial previamente ampliada, donde se identifica una amplitud de (2000-2400 mV) cada 15 segundos, con presencia de ruido. La segunda figura en la primera columna muestra la señal filtrada sobre los picos de los intervalos RR, en este caso la amplitud se encuentra entre (0-500 mV). La tercera figura de la columna izquierda representa el comportamiento del ritmo cardiaco que va desde (60-64 beats per minute). Por último, la imagen de la columna derecha indica la tasa de muestreo de la señal filtrada, es decir, genera un valor promedio del total de las muestras en un segundo, para este caso el intervalo se encuentra entre [-0.2, 0.4] entre (98-460mV).

3.Geolocalización de un vehículo para JAVA en arquitectura ARMv7l.

Cualquier actividad que precise determinar la posición de puntos, es objeto de aplicación de navegación satelital. Los Sistemas de Navegación Global por Satélite (GNSS) dentro de sus aplicaciones ofrecen la posibilidad de determinar la posición de cualquier dispositivo receptor de

señales sobre la superficie terrestre, siempre y cuando este receptor capture la señal emitida por satélites visibles. En la academia es muy común simular este tipo de sistemas para proponer alternativas de menor costo, a continuación, se describe el servicio que simula un sensor GNSS para transmisión y posicionamiento de un vehículo a través de un servidor TCP.

Implementación. El módulo GNSS (Camacho, 2017) a través de una interfaz de sensores que se despliega sobre un dispositivo elegido (tarjeta Jetson TK1) implementa una aplicación que integra información sobre la posición geográfica (latitud: longitud) de un vehículo en un mensaje con formato JSON/http. Al observar el funcionamiento de la aplicación en la plataforma, se construye la base sobre la cual se propuso la geolocalización de ambulancias.

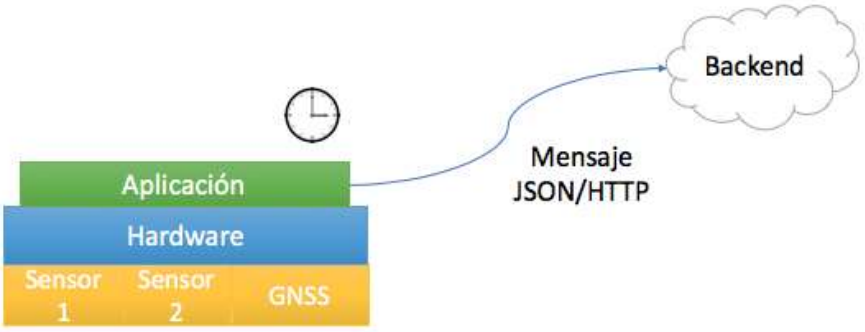

Figura 20 Aplicación GNSS

Resultados

	Embebido BRT	TCP Cliroute
T [sg]	34	23
% CPU	1.3	0.7

Figura 21 Tiempos de ejecución y %CPU utilizada

Donde:

- Embebido BRT toma los datos que vienen de los sensores, los arma en un mensaje formato JSON/HTTP y lo envía a un servidor.
- TCP Cliroute simula los sensores GNSS y en un archivo de texto muestra las líneas de salida.

6. Conclusiones

La supercomputación abrió el camino de aplicación para las plataformas embebidas, que pasaron de ser destinadas durante los últimos veinte años para propósitos generales a ser placas de desarrollo científico en áreas específicas y complejas. Por otra parte, la evolución de la GPU hasta su cuarta generación y de GPGPU produjeron su integración a la CPU y la creación de SoCs, los cuales por su bajo costo y poco consumo energético aumentan los campos de acción para desarrolladores tanto en la industria como en el sector académico. La obtención de imágenes de alta calidad en áreas específicas de la medicina a través del procesamiento de billones de operaciones en coma flotante por segundo (GFLOPS/sg) permiten a la ciencia dar un paso agigantado con aplicaciones de visualización y procesamiento de alto desempeño en áreas como la radiología y la cardiología, así como también de transmisión de datos, y por supuesto de servicios GNSS.

La abstracción de las necesidades T.I. en ambulancias fue posible en tanto se hicieron evidentes los factores que influían en la deficiente prestación del servicio, la ausencia de equipo biomédico de alto nivel de complejidad fue uno de ellos, esto justificó la implementación de la plataforma hardware y el diseño de la arquitectura software sobre la misma.

Como resultado de la integración de los componentes software, se propuso una arquitectura modular de cinco niveles en los cuales se detalla y estructura cada característica hardware de la tarjeta Jetson TK1. Cada nivel de la arquitectura software representa un paso hacia la solución propuesta para las necesidades identificadas.

Los paquetes software, librerías, dependencias y demás herramientas que se implementaron para la construcción del prototipo fueron cuidadosamente seleccionadas para cumplir funcionalmente su propósito intencional, es decir, las limitaciones en memoria y almacenamiento fueron restricciones que se tuvieron en cuenta previamente al proceso de instalación y construcción del prototipo.

La arquitectura software propuesta fue validada a través de dos etapas, un marco comparativo que pone a prueba el comportamiento de las tecnologías planteadas siguiendo el mismo parámetro y la validación a partir de tres aplicaciones independientes sobre estas tecnologías, para generar una respuesta positiva a favor de CUDA como la tecnología con mayor rendimiento y velocidad, Python con gran potencial a nivel de rendimiento, pero sin restricción de tiempo, y Java que cuenta con la mayor cantidad de aplicaciones de red, y millones de desarrolladores.

A manera de hipótesis se plantea que el producto final del prototipo propuesto en este trabajo de investigación con un 19,8% de los costos totales asumidos por prestadores de servicios de salud, ofrecería funcionalidades de igual calidad, lo que generaría mayor acceso a la población a equipos biomédicos.

Los resultados del conocimiento obtenido en este trabajo de investigación se divulgaron a través de dos eventos internacionales, Supercomputing 2016 y GPU Technology Conference 2017.

No está de más aclarar que algunas aplicaciones que requieran estructuras complejas y gran capacidad en memoria RAM, continuarán con el desarrollo sobre ordenadores convencionales, por su capacidad de producción masiva de productos más comerciales. La propuesta que se probó en este trabajo de investigación en la arquitectura ARMv7l rompe de cierta manera el paradigma, pero tiene limitaciones en memoria y almacenamiento.

7. Recomendaciones

- Desarrollar el producto software a partir del prototipo presentado, con la implantación de la plataforma en el vehículo de transporte asistencial para evaluar requerimientos funcionales como eficiencia, costo-beneficio, usabilidad, interoperabilidad, estabilidad. Y también para corroborar sus requerimientos no funcionales como seguridad, mantenibilidad, extensibilidad.
- Desarrollar aplicaciones en las áreas de e-Health y telemedicina que validen la extensibilidad de la plataforma propuesta.

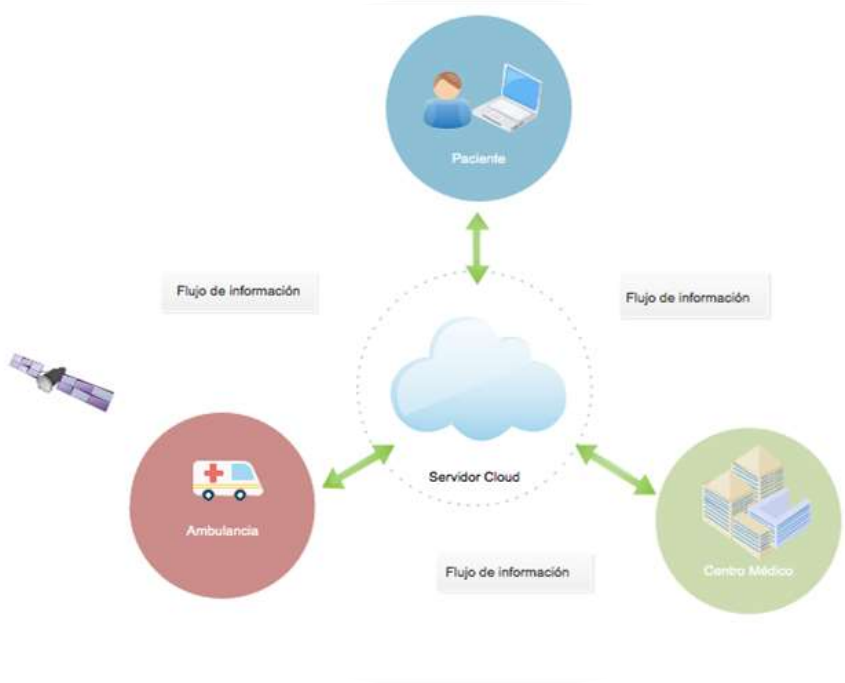

Figura 22 Sistema propuesto

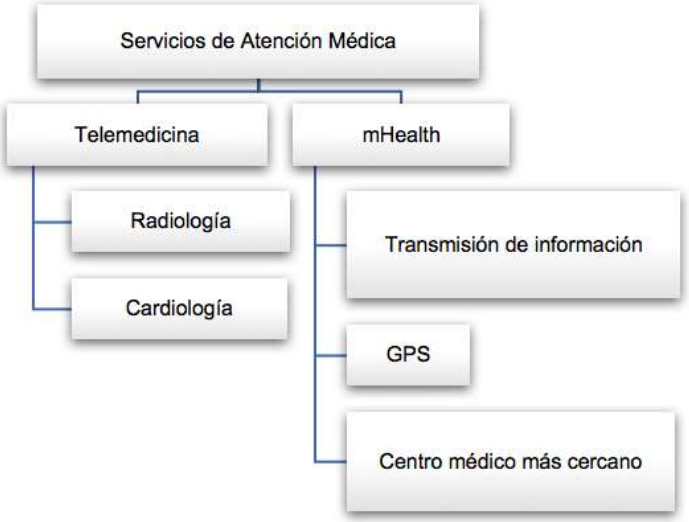

Figura 23 Trabajo futuro

Referencias Bibliográficas

A very brief history of High-Performance Computing. (2016). *Gordon College.* Recuperado el 21 de 04 de 2017, de Mathematics and Computer Science: http://www.math-cs.gordon.edu/courses/cps343/presentations/HPC_History.pdf

Abdul-Malek, A. (2014). *NVIDIA*. Obtenido de NVIDIA Blog: https://blogs.nvidia.com/blog/2015/10/02/alemhealth/

Arbeláez, J. (2008). *Widetech Colombia*. Obtenido de http://www.widetech.co/es/

Arduino. (s.f.). *Intel Edison Module*. Recuperado el 12 de 04 de 2017, de Intel Edison Module Key Features : https://www.arduino.cc/en/ArduinoCertified/IntelEdison

B. W., A. K., & J. B. (2009). Low Cost ECG Monitor for Developing Countries. *IEEE*.

Bhuy L. (INTEL). (13 de mayo de 2016). *INTEL Software*. Recuperado el 12 de 04 de 2017, de Getting to know the Arduino 101 Board.

BioSPPy Documentation. (2015). *BioSPPy*. Recuperado el 1 de 11 de 2016, de BioSPPy Docs: http://biosppy.readthedocs.io/en/stable/

biosspy.signals. (2015). *biosppy.signals*. Recuperado el 1 de 11 de 2016, de Modules: http://biosppy.readthedocs.io/en/stable/biosppy.signals.html#module-biosppy.signals.ecg

Bongi, R. (2017). *Rnext. Robotics and Automation Passion*. Recuperado el 12 de 04 de 2017, de Raffaello Bonghi Website: http://rnext.it/review/nvidia-jetson-tx2/

C. A., K. E., & Ozgul, M. E. (6 de noviembre de 2012). Demo Abstract: Telehealth - Intelligent Healthcare with M2M Communication Module. *ACM*.

C. L. (2010). *LLVM* . Recuperado el 3 de 12 de 2016, de The Architecture of Open source Applications: http://www.aosabook.org/en/llvm.html

Camacho, W. P. (30 de 1 de 2017). *Universidad Industrial de Santander*. Recuperado el 28 de 3 de 2017, de Repositorio Universidad Industrial de Santander: http://repositorio.uis.edu.co/jspui/

Carreiras, C. (2017). *GitHub*. (BioSPPY, Productor) Recuperado el 1 de 4 de 2017, de GitHuB: https://github.com/PIA-Group/BioSPPy/blob/master/examples/ecg.txt

Conda Documentation. (2015). *Conda*. Recuperado el 2 de 12 de 2016, de Conda Docs: http://conda.pydata.org/docs/index.html

D. W., & A. M. (16 de 5 de 2016). *The biggest challenges facing ambulances services*. Recuperado el 20 de 2 de 2017, de Association of Ambulance Chief Executives: http://aace.org.uk/wp-content/uploads/2016/06/HSJ-Intelligence-The-biggest-challenges-facing-ambulance-services-2016-05-16.pdf

Decreto 1471. (5 de 8 de 2014). *Régimen Legal de Bogotá D.C. , Alcaldía de Bogotá*. Recuperado el 10 de 2015, de Alcaldía de Bogotá D.C. : http://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=58845

Departamento Administrativo Nacional de Estadística. (30 de 3 de 2014). *Boletín Técnico*. Recuperado el 20 de 1 de 2017, de Estadísticas Vitales Cifras Preliminares 2014-2015: https://www.dane.gov.co/files/investigaciones/poblacion/bt_estadisticasvitales_2014p-2015p-30-03-2016.pdf

Digi-Key Corporation . (s.f.). *Digikey* . (B. Maker.io, Productor) Recuperado el 12 de 04 de 2017, de Platforms: https://www.digikey.com/en/maker/blogs/intel-how-to-choose-the-right-board-for-your-project/de40a273578844e3a9b64f64c5679fdf

Digisens. (2016). *Digisens*. Recuperado el 20 de 04 de 2017, de Health care Imaging: http://www.digisens3d.com/en/

E. B., & J. S. (4 de 1993). The emergent problem of ambulance misuse. *US National Library of Medicine*.

evolutivo, C. d. (2006). *Prototipado Evolutivo.* Recuperado el 28 de 1 de 2017, de Requisitos Software: https://requisitosdesoftware.wikispaces.com/D.+Prototipado+Evolutivo?responseToken=eee805b2dcf977f816adfc8f95c8b80a

F. A., M. A., & M. A. (2008). A pruned Fuzzy K-Nearest Neighbor Classifier with Application to Electrocardiogram Based Cardiac Arrhytmia Recognition. *IEEE*.

F. S., T. G., T. K., K. B., & S. A. (2008). A connectivity management system for vehicular telemedicine applications in heterogeneous network. *ACM*.

Federal Specification for the Star-of-Life Ambulance. U.S. . (1 de 8 de 2007). *General Service Administration*. Recuperado el 10 de 2015, de KKK-A-1822F: http://www.emt-resources.com/support-files/kkk-a-1822f-08.01.2007.pdf

General Services Association. (2010). *General Services Association*. Recuperado el 10 de 2015, de Policy & Regulations: https://www.gsa.gov/portal/category/100000

Gordon College. (2016). *Gordon College* . Recuperado el 21 de 04 de 2017, de Mathematics and Computer Science: http://www.math-cs.gordon.edu/courses/cps343/presentations/HPC_History.pdf

Ingeniería en Sistemas. (2009). *Universidad Nacional de Loja.* Recuperado el 10 de 2015, de Prototipos Informáticos: https://sistemas2009unl.wordpress.com/prototipos-informaticos/

INTEL . (s.f.). *Intel Developer Zone*. Recuperado el 12 de 04 de 2017, de Internet of Things: https://software.intel.com/es-es/iot/hardware/joule

INTEL. (2014). *INTEL Galileo Gen 2 Development Board*. Recuperado el 12 de 04 de 2017, de INTEL Galileo Gen 2 Datasheet: http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/galileo-g2-datasheet.pdf

INTEL. (abril de 2017). *INTEL*, 1.5. Recuperado el 12 de 04 de 2017, de Intel Joule Module Datasheet: http://www.intel.com/content/dam/support/us/en/documents/joule-products/intel-joule-module-datasheet.pdf

J. S., & M. S. (2 de 11 de 2009). Activity-Aware ECG-based Patient Authentication for Remote Health Monitoring. *ACM*.

Jain, P. (marzo de 2016). *Engineers Garage*. Obtenido de Embedded System: https://www.engineersgarage.com/articles/embedded-systems

La Web del Electrocardiograma. (2013). *My EKG*. Recuperado el 5 de 12 de 2016, de Intervalos y Segmentos del Electrocardiograma: http://www.my-ekg.com/generalidades-ekg/intervalos-segmentos-ekg.html

LLVM. (2011). *The LLVM Compiler Infraestructure*. (C. S. Department, Productor) Recuperado el 1 de 12 de 2016, de LLVM Overview: http://llvm.org/

LLVM. (2016). *LLVM Downloads*. Obtenido de http://releases.llvm.org/download.html

M. M., S. H., & H. T. (2009). Ambulance Redeplyment: An approximate dynamic for programming approach. *IEEE*.

Miniconda. (2016). *Continuum Miniconda*. Obtenido de https://www.google.com/url?q=http%3A%2F%2Frepo.continuum.io%2Fminiconda%2FMiniconda-latest-Linux-armv7l.sh&sa=D&sntz=1&usg=AFQjCNGkbgPc-AVNUudoite3fvPqdJU2QQ

Ministerio de Salud y Protección Social. (2012). *Publicaciones.* Recuperado el 9 de 12 de 2016, de Guías Básicas de Atención Médica Prehospitalaria.: https://www.minsalud.gov.co/Documentos%20y%20Publicaciones/Guias%20Medicas%20de%20Atencion%20Prehospitalaria.pdf

Ministerio de Salud y Protección Social. (2013). *Instituto Nacional de Vigilancia de Medicamentos y Alimentos INVIMA.* (M. R. Zapata, Ed.) Recuperado el 10 de 2015, de ABC De Dispositivos Médicos: https://www.invima.gov.co/images/pdf/tecnovigilancia/ABC%20Dispositivos%20Medicos%20INVIMA.pdf

Ministerio de Salud y Protección Social. (2014). *Normatividad.* Recuperado el 10 de 2015, de Resolucion 00002003 de 2014 :

https://www.minsalud.gov.co/Normatividad_Nuevo/Resoluci%C3%B3n%202003%20de%202014.pdf

Ministerio de Salud y Protección Social. (11 de 2015). *Análisis de Situación de Salud. Colombia.* Obtenido de Dirección de Epidemiología y Demografía: https://www.minsalud.gov.co/sites/rid/Lists/BibliotecaDigital/RIDE/VS/ED/PSP/asis-2015.pdf

Moreno, A. (2016). *Ambulance to Go*. Obtenido de http://www.e-takeoff.com/ambulancetogo/

N. S., & B. R. (2016). Probabilistic Study and embedded system. *IEEE*. Obtenido de http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7804988

National EMS Assessment . (2011). *Federal Interagency Committee on Emergency Medical Services.* (M. B. Greg Mears, Ed.) Recuperado el 10 de 2015, de https://www.ems.gov/pdf/2011/National_EMS_Assessment_Final_Draft_12202011.pdf

NFPA 1917. (2015). *National Fire Protection Association.* Recuperado el 10 de 2015, de Codes and Standards: http://www.nfpa.org/codes-and-standards/all-codes-and-standards/list-of-codes-and-standards?mode=code&code=1917

Norma Técnica Colombiana. (12 de 12 de 2007). *Norma Técnica Colombiana NTC 3729.* Obtenido de Tipología Vehicular. Ambulancias de Transporte Asistencial Básico : http://repository.unac.edu.co/jspui/bitstream/11254/160/20/Norma%20t%C3%A9cnica%20Colombiana%203729%20ambulancias

Numba . (2015). *Numba Overview*. Obtenido de Numba documentation: http://numba.pydata.org/numba-doc/0.29.0/user/overview.html

NVIDIA . (2014). *NVIDIA Embedded Systems*. Recuperado el 12 de 04 de 2017, de Modules and Developments Kits: http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html

NVIDIA . (2014). *NVIDIA Jetson*. Recuperado el 12 de 04 de 2017, de Embedded Systems Developer kits Module: http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html

NVIDIA. (2014). *NVIDIA Tegra Processors*. Recuperado el 12 de 04 de 2017, de TK1 Processor Specifications: http://www.nvidia.com/object/tegra-k1-processor.html

NVIDIA Embedded Computing. (9 de 2014). *Tools*. Obtenido de Jetpack: https://developer.nvidia.com/embedded/jetpack

Oracle. (2017). *Java SE Development Kit Downloads*. Obtenido de http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Piñeros, R. R. (2015). *Informe de ponencia para primer debate al Proyecto de Ley 95 de 2015 Senado, 249 de 2016 Cámara.* Comisión Séptima Constitucional Permanente , Cundinamarca, Bogotá D.C. Obtenido de http://www.saludcapital.gov.co/Documents/NormatividadenSalud/Octubre%202016/14%20INFORME%20DE%20PONENCIA%20PARA%20PRIMER%20DEBATE%20AL%20PROYECTO%20DE%20LEY%2095%20DE%202015%20SENADO.pdf

Plan de Desarrollo Departamental, Suárez . (22 de agosto de 2016). *Ministerio de Salud y Protección Social.* Recuperado el 20 de 01 de 2017, de Proyectos TIPO: https://proyectostipo.dnp.gov.co/images/pdf/ambulancias/MGAambulancias.pdf

Proyecto Estándar. (2015). *Ministerio de Salud y Protección Social* . Recuperado el 12 de 2016, de Proyectos TIPO: http://viva.org.co/PDT_para_la_Construccion_de_Paz/Proyectos_tipo_SGR-DNP/Estandarizado%20Ambulancias%20V4.pdf

Proyectos TIPO. (Julio de 2016). *Ministerio de Salud y Protección Social.* Recuperado el 8 de 11 de 2016, de Dotación De Ambulancias; Transporte Asistencial Básico y Medicalizado: https://proyectostipo.dnp.gov.co/images/pdf/ambulancias/PTambulancias.pdf

Python Software Foundation. (2016). *Matplotlib*. Obtenido de https://pypi.python.org/pypi/matplotlib

Python Software Foundation. (2016). *Virtualenv*. Obtenido de https://pypi.python.org/pypi/virtualenv

S. I. (2010). *Software Engineering 9* (9ª Edición ed.). (M. Hirsch, Ed.) Adisson-Wesley.

T.-H. T., C.-S. C., & C. L. (septiembre de 2009). Development of a Portable Linux-Based ECG Measurement and Monitoring System . (L. 2. Septiembre 2009. Springer Science and Business Media, Ed.)

TechniScan . (2016). *NVIDIA*. Recuperado el 10 de 04 de 2017, de GPU Applications : http://www.nvidia.com/object/imaging.html

Universidad del País Vasco. (2010). *Universidad del País Vasco UPV*. Recuperado el 2 de 12 de 2016, de La Máquina Virtual Java: http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/introduccion/virtual.htm

V. S., & K. D. (12 de 2010). Ambulance location and relocation problems with time-dependent travel times. (E. J. Research, Ed.) *SienceDirect, 2017*(3), 1293.

World, A. o. (25 de april de 2015). *Worcester Polytechnic Institute.* Obtenido de Worcester Polytechnic Institute: https://web.wpi.edu/Pubs/E-project/Available/E-project-042413-092332/unrestricted/MQFIQP2809.pdf

Y. L., Y. Z., S. J., W. W., L. H., & Z. G. (06 de 06 de 2015). Location Selection for ambulance stations: a data-driven approach. *ACM*.

Apéndices

Apendice A. Construcción del prototipo. Tecnologías Java y Python.

5.4 Construcción del prototipo

5.4.2. JAVA

Instalación del JDK:

1. Descarga del JDK 8u45 en el sitio web oficial (Oracle, 2017).

- Linux ARM 32 Hard Float ABI
- Arquitectura ARMV7vl de 32 bits (NVIDIA Jetson TK1)
- Configuración de las preferencias del compilador y la máquina virtual

```
/usr/local/java$ sudo update–alternatives --set java
/usr/local/java/jdk1.8.0_111/bin/java

/usr/local/java$ sudo update–alternatives --set javac
/usr/local/java/jdk1.8.0_111/bin/javac
```

2. En la terminal se ven la versión de la máquina virtual y del compilador a través de:

```
/netbeans-8.0.1/java$ java –version
```

```
java version “1.8.0_111”
Java (TM) SE Runtime Environment (build 1.8.0_111-b14)
```

3. Se reinicia el sistema

Instalación de Netbeans:

1. Descarga desde el sitio web (Oracle, 2017)

- Plataforma Linux
- Versión 8.0.2
- Paquete de Descarga: Java SE (NetBeans Platform SDK, Java SE, Java FX).

2. Se inicia desde la ruta que contiene el archivo ejecutable “netbeans”:

```
/netbeans-8.0.1/bin$ ./netbeans
```

5.4.3. Python

Instalación de Miniconda:

1. Descarga desde el sitio web (Miniconda, 2016)

- Linux armv7l Installer
- Versión 3.18.3

```
$ conda update conda
$ conda install accelerate
```

2. Se listan paquetes en Miniconda

```
$ conda list
```

Nota: Durante la instalación de Miniconda 3.18.3 para la versión de Python 2.7.10 en la tarjeta NVIDIA Jetson TK1 los paquetes no incluían Numba, por lo que fue necesario realizar su instalación con la herramienta pip.

Instalación de pip:

1. Se ejecuta el siguiente comando:

```
$ conda install pip
```

2. Verificación versión de pip 9.0.0

```
$ pip --version
```

```
pip 9.0.0 from  /home/ubuntu/miniconda/lib/python2.7/site-packages
(python 2.7)
```

Instalación LLVM (LLVM, 2016):

Requisitos:

- GNU Make
- GCC
- Python
- zlib

1. Se instalan los paquetes que permitirán la instalación de llvmlite, y se cambia la dirección en el archivo de configuración $PATH

```
$ [sudo] apt-get install zlib  zlib1g zlib1g-dev libedit2 libedit-dev llvm-3.6 llvm-3.6-dev llvm-dev
pip install enum34 funcsigs

LLVM_CONFIG=/usr/bin/llvm-3.6 pip install llvmite
```

2. Actualización de paquetes en Miniconda

```
$ conda update conda
$ conda upgrade conda
```

3. Instalación global de virtualenv (Python Software Foundation, 2016) (Versión pip≥1.3)

```
$ [sudo] pip install virtualenv
```

4. Instalación del paquete python-dev

```
$ [sudo] apt-get install python2.7-dev
```

Instalación de Numba:

Requisitos:

- llvmlite armhf
- Numpy
- Funcsigs
- Enum34

1. A través de pip

```
$ pip install numba
```

2. Verificación versión numba 0.29.0

```
$ pip show numba
```

Name: numba
Version: 0.29.0
Summary: compiling Python code using LLVM
Home-page: http://numba.github.com
Author-email: numba-users@continuum@io
License: BSD
Location: /home/ubuntu/miniconda/lib/python2.7/site-packages
Requires: singledispatch, enum34, llvmlite, funcsigs, numpy

$ pip --version

$ pip 9.0.0 from /home/ubuntu/miniconda/lib/python2.7/site-packages (python 2.7)

Apéndice B. Poster presentado en el workshop: Women in High Performance Computing (WHPC). Supercomputing 2016 (SC16). Salt Lake City, Utah, Estados Unidos. Noviembre 13 de 2016

High performance embedded computing platform for emergency vehicle transportation

Maria Juliana Garzón Vargas, Gabriel Pedraza Ferreira, Carlos J. Barrios Hernández,
High Performance and Scientific Computing Center
Universidad Industrial de Santander, Bucaramanga, Colombia.

Objectives

- Identify the IT needs for emergency vehicles
- Propose an extensible software architecture
- Develop the platform's functional prototype

Background and Motivation

Basic life support in terms of Emergency Medical Services is the action of taking patients from the place where the event occurred to the closest medical center. This directly involves pre-hospital healthcare, which at the same time relies on a set of IT technologies that aim to improve patient care, being this the very first contact of patients with the health care system.

However, the technological support previously mentioned suggests a high cost for all EMS' entities, not only with the purchase of it but the continuing technical assistance this devices require, which creates a gap in the service as it is not available worldwide, as a consequence, a low-cost alternative solution is required.

All in all, it's appropriate to propose that this kind of support can be justified by the assessment and diagnosis of the patient's state along the way with the transmission of pertinent information to the medical center. Through a platform that improves existing features with optimal use of resources this goal will be achieved.

Methods

The software development methodology implemented to introduce the platform is an adaptation of the evolutive prototyping methodology. This research is currently working in phase fourth of the model.[1]

1. Needs identification
2. Architecture specification
3. Building the prototype
4. Fittings and validation on Jetson TK1

EMS worldwide

To understand the first phase and how the global emergency medical service systems work nowadays, we focus on three targets:

- EU: CEN 1789:2007 [2]
- USA: KKK-A-1822F[3], NFPA 1917 [4]
- COL: NTC 3729–2007 [5]

Hardware Platform NVIDIA Jetson TK1

We have chosen the embedded hardware architecture NVIDIA Jetson Tegra Kepler 1, since it provides the right performance features and requirements for this matter; it's positioned as the first supercomputer for embedded systems and is chosen for the development of high-quality visual applications against a considerable level of reliability. [6]

Software Architecture

So far, the outcome is the model of a software architecture composed of three layers: **1. Hardware Layer**, according to the Jetson TK1 hardware **2. Software Layer**, composed of Middlewares (Low Level Virtual Machine and Java Virtual Machine), Operating System (Ubuntu 14.04) and Device Drivers **3. Application Layer**, where the main goal is running three medicine applications: two HPC apps and one generic, within its respectively libraries (i.e. for CUDA: cuFFT, Arrayfire, OpenCV). [7]

Further work

Applications developed on the last layer of the platform from telemedicine to additional services as GPS localization and transmission of relevant information are thought to be the future work for this research.

References

[1] Technische Universität München, Department of Informatics, Robotics and Embedded Systems. Website: www6.in.tum.de/Main/ResearchFtos
[2] AENOR. Website: www.aenor.es/aenor/normas/normas/fichanorma.asp?tipo=N&codigo=N0045721#.V_aaR4_hDct
[3] U.S. General Services Administration, Federal Specification for the Star-of-Life Ambulance. Website: www.ok.gov/health2/documents/KKK-A-1822F%20%2007.01.2007.pdf
[4] National Fire Protection Association, Proposed Draft of NFPA 1917, Standard of Automotive Ambulances 2013 Edition. Website: www.nfpa.org/Assets/files/AboutTheCodes/1917/NFPA1917Draft.pdf
[5] NTC 3729-2007. Website: repository.unac.edu.co/jspui/bitstream/11254/160/20/Norma%20t%C3%A9cnica%20Colombiana%203729%20ambulancias
[6] NVIDIA Jetson TK1. Embedded Systems (2014). Website: www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html
[7] Noergaard, T. (2005), Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers, Elsevier Editorial.

Super Computación y Cálculo Científico UIS
GPU RESEARCH CENTER
GPU EDUCATION CENTER

Apéndice C. Poster presentado en 2017 GPU Technology Conference (GTC17). San José, California, Estados Unidos. Mayo 08 de 2017.