DESARROLLO DE UN PROTOTIPO FUNCIONAL DE BAJO COSTO PARA LA MEDICIÓN DE LA SEVERIDAD VIBRATORIA EN MÁQUINAS ROTATIVAS SEGÚN LAS NORMAS ISO 2372 Y 10816-3

JAIME ALBERTO SERNA GALLO

UNIVERSIDAD EAFIT
ESCUELA DE INGENIERÍAS
MAESTRÍA EN INGENIERÍA
ÁREA DE MANTENIMIENTO INDUSTRIAL
MEDELLÍN
2012

DESARROLLO DE UN PROTOTIPO FUNCIONAL DE BAJO COSTO PARA LA MEDICIÓN DE LA SEVERIDAD VIBRATORIA EN MÁQUINAS ROTATIVAS SEGÚN LAS NORMAS ISO 2372 Y 10816-3

JAIME ALBERTO SERNA GALLO

Proyecto de maestría para optar al título de:

Magister en Ingeniería

Tutor:

Dr. Ing. Leonel Francisco Castañeda Heredia

UNIVERSIDAD EAFIT
ESCUELA DE INGENIERÍAS
MAESTRÍA EN INGENIERÍA
ÁREA DE MANTENIMIENTO INDUSTRIAL
MEDELLÍN
2012

AGRADECIMIENTOS

A mi familia…

TABLA DE CONTENIDO

pág.

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

TABLA DE ANEXOS

pág.

GLOSARIO

ADC (ANALOG-DIGITAL CONVERSOR): módulo de hardware encargado de convertir valores análogos en valores discretos.

ABSTRACCIÓN: el término se refiere al énfasis en el "¿qué hace?" más que en el "¿cómo lo hace?" (Característica de caja negra).

API (APPLICATION PERIHELIA INTERFACE): es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

ARM: Advanced RISC Machines. Procesadores de última generación de gran eficiencia y relativo bajo costo.

ASIC (APPLICATION-SPECIFIC INTEGRATED CIRCUIT): es un circuito integrado hecho a la medida para un uso en particular, en vez de ser concebido para propósitos de uso general.

DSP (DIGITAL SIGNAL PROCESSOR): abreviatura de, es un dispositivo de procesamiento optimizado para el trabajo con señales digitales, posee algunas características como multiplicadores en paralelo y manejo de coma flotante.

DSC (DIGITAL SIGNAL CONTROLLER): se denominan así los dispositivos procesamiento tales como los microcontroladores que poseen algunas características de los DSP pero que siguen conservando su bajo costo.

EEPROM (ELECTRICALLY ERASABLE PROGRAMMABLE READ ONLY MEMORY): es un tipo de memoria que almacena datos permanentemente y que puede ser borrada y reprogramada a través de campo eléctrico.

Full-Duplex: se dice del protocolo de comunicación donde los datos se transmiten bidireccionalmente por dos líneas y es posible realizar transmisión y recepción al mismo tiempo.

Half-Duplex: se dice del protocolo de comunicación donde los datos se transmiten bidireccionalmente por una sola línea y que la transmisión y la recepción no puede darse al mismo tiempo.

I/O: Abreviatura para Input/Output o entrada/salid, son las interfaces entre un procesador y el mundo.

I2C: abreviatura para Inter-Integrated Circuit, es un protocolo de comunicaciones usado en dispositivos electrónicos para la comunicación entre periféricos.

LED (LIGHT-EMITTING DIODE): Dispositivo semiconductor que radia luz cuando es polarizado, su función sigue siendo la de transmitir la corriente cuando esta circula en un solo sentido.

NETLIST: es un archivo de texto en ASCII que posee todas las características de cada componente con sus pines y a donde está conectado cada uno de ellos, cada conexión es llamada nodo.

SPI: Serial Perihelial Interface, protocolo de comunicaciones utilizado para la comunicación de micro controladores con sus periféricos.

STAND-ALONE: Software de computador que puede trabajar sin conexión, esto es que no requiere una conexión a una red para funcionar.

Time To Market: tiempo total de calendario requerido para llevar un producto desde la concepción hasta el mercado.

Timer: es un periférico que mide lapsos de tiempo, típicamente contando los ciclos de procesamiento o reloj del sistema.

UML: Abreviatura de Unified Modeling Language, Lenguaje de modelado de sistemas de información a través de diagramas de flujo.

RESUMEN

Actualmente el mantenimiento predictivo es muy poco usado en la pequeña y mediana empresa (Pymes), esto se debe al costo de implementación, equipos y personal capacitado en esta área, lo cual desaprovecha las ventajas que trae este tipo de mantenimiento, que a pesar del costo, trae beneficios económicos a largo plazo tales como aumentar la vida útil de la maquinaria y disminuir los costos por reparación.

Hoy en día los avances de los sistemas electrónicos avanzan a pasos gigantescos, ofreciendo dispositivos cada vez con mayores prestaciones y a muy bajo costo, lo cual debería verse reflejado en la disminución del valor de los equipos de mantenimiento, situación que no sucede y que se debe a la falta de aprovechamiento y apropiación tecnológica que posee el país.

Como primer paso en la búsqueda de desarrollar equipos de bajo costo que ayuden a aprovechar las ventajas y bondades que ofrece el mantenimiento predictivo, se realizó una revisión del estado del arte de las metodologías de diseño de sistemas embebidos, y con base en dicha revisión, se generó un protocolo de desarrollo específico para equipos electrónicos en esta área.

Una vez realizado el protocolo, se desarrolló un sistema de bajo costo para la evaluación de la severidad vibratoria en máquinas rotativas que solo requiere la presencia de un experto en vibraciones la primera vez que se instala, posterior a esto, el usuario de la empresa puede seguir monitoreando por cuenta propia sus equipos y generar acciones de mantenimiento solo cuando sea necesario, de este modo queda demostrado que si es posible realizar implementaciones de bajo costo y que el protocolo desarrollado es una buena herramienta para generar

muchos más equipos de este tipo que beneficien a la sociedad y al desarrollo económico del país.

Palabras clave: sistemas embebidos, mantenimiento predictivo, protocolo desarrollo.

INTRODUCCIÓN

Este trabajo toma como problemática inicial, el alto costo de los sistemas de monitoreo para el mantenimiento predictivo lo cual genera la poca acogida de esta metodología de mantenimiento en los sectores de poco poder adquisitivo tal como lo son las pequeñas y medianas empresas (Pymes) del país, de este modo se desaprovecha las ventajas que trae este tipo de mantenimiento, que a pesar del costo, trae beneficios económicos a largo plazo tales como aumentar la vida útil de la maquinaria y disminuir los costos por reparación.

Con base en esta problemática, se propone la realización de un sistema de bajo costo que proporcione una herramienta eficaz para la implementación de este tipo de mantenimiento.

Hoy en día los sistemas embebidos están presentes en casi cualquier sistema electrónico de uso común, si bien su espacio de aplicación es muy amplio, no existe una metodología de desarrollo bien defina que describa los diferentes paso que hay que realizar para desarrollar un dispositivo de este tipo.

Dado este inconveniente se hace inicialmente una revisión bibliográfica de los modelos y metodologías de desarrollo de sistemas embebidos, posterior a eso, se define un protocolo de desarrollo que reúne los mejores aspectos de las diferentes metodologías, y se crea un esquema de desarrollo a partir del cual sea posible generar sistemas electrónicos para el monitoreo de sistemas y herramientas de mantenimiento predictivo.

Una vez se genera el protocolo de desarrollo, se siguen sus etapas para obtener un sistema de monitoreo sencillo pero muy útil a la hora de implementar en la

industria, en este documento se describen los diferentes paso de la metodología y los resultados obtenidos durante la aplicación del protocolo.

1. PLANTEAMIENTO DEL PROBLEMA

Según el gobierno nacional establece como micro, pequeña y mediana empresa, toda unidad de explotación económica, realizada por persona natural o jurídica, en actividades empresariales, agropecuarias, industriales, comerciales o de servicio, rural o urbana que cumpla con los criterios de activos y número de empleados para pequeña y mediana que se presentan en la Tabla 1. [Rodríguez, 2003]

Tabla 1. Clasificación de PYME en Colombia

Tipo de empresa	**Número de empleados**	**Activos**	**Rangos de Activos Empresa en Pesos**
Micro	Hasta 10	Menos de 501 SMLV	Menos de $166.332.000
Pequeña	Entre 11 y 50	Desde 501 y menos de 5.001 SMLV	Entre $166.332.000 y $1.660.332.000
Mediana	Entre 51 y 200	Desde 5.001 y menos de 15.000 SMLV	Más de $1.660.332.000 y $4.980.000.000
Grande	Más de 200	Más de 15.000 SMLV	Más de 4.980.000.000

Rodríguez, 2003

De acuerdo con las cifras del Dane (Departamento Administrativo Nacional de Estadística) para el año 2005, las microempresas conforman el 96,4% de los establecimientos, las pequeñas el 3,0%, las medianas el 0,5%, y las grandes 0,1%. Por personal ocupado, las microempresas representan el 50,3% del empleo, las pequeñas el 17,6%, las medianas el 12,9%, y las grandes el 19,2%. [Misiónpyme, 2008], de este modo queda claro que a pesar de que las PYME son empresas en proceso de crecimiento, representan un gran porcentaje de la

economía del país y que a su vez, la aplicación de la tecnología que mejoren sus procesos de producción, se verán reflejadas en el aumento del desarrollo industrial del país, un alto porcentaje de pequeñas y medianas empresas se dedica a una gran variedad de actividades del sector servicios (59.2%). El comercio y la industria le siguen en importancia con 21.5% y 19.3%, respectivamente. [Muñoz, 2009]

Actualmente en el país, la industria electrónica se dedica a la implementación de soluciones a las necesidades de los clientes a partir de elementos existentes importados desde el exterior, y la poca investigación y desarrollo que se hace, está enfocada en la solución de problemas puntuales de una empresa y no al desarrollo como tal de esta industria, apenas hasta hace pocos años empezaron a surgir algunas empresas dedicadas a la fabricación de tarjetas electrónicas, que aún no están en capacidad de producir elementos de alta complejidad tales como tarjetas electrónicas varias capas, brindando así un aporte casi nulo al mercado electrónico mundial, que hoy en día es dominado en su mayoría por el mercado asiático.

Por otro lado, se observa que en el sector industrial manufacturero del país, están involucradas una serie de maquinarias que son de vital importancia para la operación de cada compañía, debido a que repetidas fallas de estos equipos llevan a grandes pérdidas de producción y afectaciones económicas, se hace necesario que las empresas implementan diversas acciones de mantenimiento para contrarrestar el desgaste normal que sufren las máquinas durante su funcionamiento, y evitar así la perdida de utilidad por estados inoperantes.

Entre estas acciones se encuentran:

- El mantenimiento reactivo: en donde la máquina opera hasta que entra en fallo, y una vez alcanza ese estado es reparada o reemplazada.

- El mantenimiento preventivo: basado en el tiempo donde periódicamente se realiza una revisión de la máquina y se corrigen las fallas que se detecten.

- El mantenimiento predictivo basado en la condición: en donde se evalúa la condición mecánica de la máquina y su evolución, mientras está funcionando, a través de diversos síntomas que ella emite al exterior. Con base en esto se programan las necesidades de mantenimiento cuando se detecta un problema. [Saavedra, 2000]

Esta última estrategia de mantenimiento pose un costo mayor que las demás, pero de igual forma trae beneficios tales como abaratar los procesos de mantenimiento y aumentar la vida útil de los equipos, reduciendo el tiempo entre compras de maquinaria, programando los mantenimientos solo cuando se ha detectado un síntoma cuya evolución desembocará en un fallo, de igual forma en este punto los costos de reparación son mucho menores, hay que realizar un menor número de compras y se puede tener un menor inventario de repuestos en bodega. [Inessman]

A pesar de todos estos beneficios el problema de la falta de aplicación de las estrategias de mantenimiento, se debe al alto costo de estas tecnologías, existen diferentes actividades para aplicar estrategias de mantenimiento predictivo tales como análisis de vibraciones, análisis posterior de fallas, alineaciones de precisión, balanceo dinámico de rotores, ensayos no destructivos, termografía, entre otras, el costo de estas es bastante alto, lo cual no le permite a una gran cantidad de empresas acceder fácilmente a estos equipos, razón por la cual optan por estrategias de mantenimiento más pasivas y más baratas, que a largo plazo disminuyen la vida útil de sus máquinas.

La mayor cantidad de fallas que se registran en una empresa, están asociadas a los componentes mecánicos, debido a que este tipo de componentes sufren desgaste durante su ciclo de operación, lo cual produce holguras mecánicas, desbalanceos que producen vibración, o rozamiento mecánico entre elementos del sistema, este tipo de síntomas si no es corregido a tiempo pueden llevar a un rápido degradamiento de las máquinas, un método de detección de esos síntomas, es a través del análisis de vibraciones mecánicas, a partir de los cuales, se identifican patrones de frecuencias que muestran la falla del sistema.

El problema es que el costo de estos equipos es muy alto por lo cual la mayoría de PYMES no están en capacidad de comprarlos, en el ANEXO A se encuentra una tabla comparativa de algunos equipos de medición de vibración en donde el más barato tiene un costo cercano a los treinta millones de pesos ($ 30.000.000), cifra que puede ser equivalente al costo de toda la maquinaria de una microempresa y solo por un equipo de vibraciones. [Cemb-Hofmann, *Emerson* process, Pruftechnik, Semapi]

Actualmente con los desarrollos tecnológicos y la implementación de procesos de producción en masa, los componentes electrónicos son de relativo bajo costo, lo que implica que el precio de producción de este tipo de equipos de monitoreo no sea tan alto. El problema del costo radica en que los fabricantes se aprovechan en que las metodologías de diseño y el conocimiento de producción de este tipo de equipos no es muy común en el medio Colombiano, por lo cual obtienen mayores beneficios, sacando mayores ganancias a partir de la venta de sus productos sin necesidad que el costo real sea ese.

Este proyecto pretende realizar una apropiación de la tecnología de diseño de sistemas embebidos portables para la medición de vibraciones, de modo que sea posible realizar implementaciones de bajo costo que permitan a las empresas el acceso a estas estrategias de mantenimiento con equipos que permitan aumentar

la vida de su maquinaria y así mejorar su competitividad en el medio y le permitan obtener un mayor beneficio económico, de igual forma se busca brindar una herramienta a partir de la cual sea posible contribuir con la aplicación de normas para la evaluación de equipos.

2. HIPÓTESIS DE LA INVESTIGACIÓN

Esta investigación busca desarrollar un protocolo a partir del cual sea posible realizar proyectos de construcción de productos electrónicos de modo que se reduzcan al máximo los tiempos de desarrollo y producción, teniendo en cuenta los períodos de gestión de componentes, evitando los tiempos muertos generados a partir de errores de diseño, adquisición de componentes y sub-dimensionamiento en la elección de los componentes.

Como paso adicional se busca construir un sistema embebido portable para la medición de vibraciones siguiendo el protocolo propuesto, de este modo se puede validar el método de diseño y a la vez obtener un resultado que reafirme el trabajo realizado.

3. OBJETIVOS

3.1. GENERAL

Desarrollar un equipo prototipo funcional de bajo costo para la medición de la severidad vibratoria en máquinas rotativas según las normas ISO 2372 e ISO 10816-3, orientado a las necesidades de la pequeña y mediana empresa Colombiana.

3.2. ESPECÍFICOS

- Revisar el estado de arte en sistemas embebidos.
- Seleccionar un método para el diseño de sistemas embebidos de bajo costo
- Elaborar el protocolo de desarrollo con base en el método previamente elaborado.
- Construir un prototipo de sistema embebidos de bajo costo para la medición de la severidad vibratoria en máquinas rotativas.

4. SISTEMAS EMBEBIDOS

A continuación se hace una reseña sobre los sistemas embebidos, se inicia con algunas definiciones y luego se muestra el inicio y las tendencias de estos dispositivos, así como las oportunidades y desafíos que se encuentran en este momento, de este modo, en la siguiente sección, será posible incursionar de mejor manera en los métodos de desarrollo de estos sistemas, por último en esta sección se hace una pequeña revisión de las aplicaciones de los sistemas embebidos para el análisis de severidad vibratoria, tema en el cual se realizara el desarrollo como método de evaluación del protocolo propuesto en la siguiente sección.

4.1. DEFINICIÓN

En la bibliografía existente sobre sistemas embebidos (SE) se encuentran una gran variedad de definiciones, por ejemplo el autor Marwedel [2003], expresa que un SE es un sistema de procesamiento de información de uso específico integrado en otro sistema de mayor tamaño y conformado por componentes hardware y software, por otro lado el autor Ganssle [2003] expresa que es una combinación de hardware, software y partes mecánicas o quizá más, destinados a desempeñar una función específica. En algunos casos, los sistemas embebidos forman parte de un sistema mayor o producto, como en el caso de un sistema de frenos antibloqueo en un carro en contraste con computadora de propósito general.

La definición de un sistema embebido es fluida y difícil de precisar debido a los constates avances tecnológicos y el dramático decremento en el costo de implementación de los componentes de hardware y software. En los últimos años, este campo se ha quedado pequeño en sus descripciones tradicionales y se

encuentran muchas variaciones de su definición, a continuación se muestran algunas y la descripción del porque se han quedado pequeñas.

"Los sistemas embebidos están más limitados en funcionalidades de software y hardware que una computadora personal (PC)": esta definición en la actualidad es solo parcialmente verdadera, porque las tarjetas y el software que se encuentran típicamente en un computador del pasado y el presente, han sido reemplazados en diseños de sistemas embebidos más pequeños y complejos, tales como dispositivos personales, PDA, y demás elementos que poseen la capacidad de procesamiento y funciones de computación, en dispositivos que caben en la palma de la mano.

"*Un sistema embebido es diseñado para desempeñar una tarea específica*": La mayoría de los sistemas embebidos son sistemas que han sido diseñados primariamente para funciones específicas, sin embargo, ahora se ven dispositivos tales como asistentes personales de datos (PDA) o celulares híbridos, los cuales son sistemas embebidos diseñados para estar en capacidad de hacer una variedad de funciones, también los últimos sistemas de televisión, incluyen aplicaciones interactivas que tienen funciones no relativas a la televisión, tales como e-mail, navegación web y juegos.

"*Un sistema embebido es un sistema computacional con requerimientos de mayor calidad y fiabilidad que otros tipos de sistemas computacionales*": es cierto que algunas familias de dispositivos integrados requieren un umbral muy alto de calidad y fiabilidad, por ejemplo si el controlador del motor de un carro falla mientras se conduce en un una autopista muy ocupada o si un dispositivo medico critico posee un malfuncionamiento en medio de una cirugía, producirán graves problemas, sin embargo algunos dispositivos, tales como sistemas de TV, juegos y celulares que funcionen mal, producirán un inconveniente, pero no serán una situación potencialmente mortal.

"*Algunos dispositivos que son llamados sistemas embebidos tales como PDAs o Web pads, no son sistemas embebidos*": Existe una cierta discusión sobre si los nuevos sistemas que cumplen con algunos, pero no todas las definiciones tradicionales del sistema embebido se clasifican o no dentro de estos, algunos consideran que la designación de estos diseños más complejos, tales como PDAs, como sistemas embebidos es impulsada por profesionales no del marketing y ventas, en lugar de ingenieros técnicos, en realidad, los ingenieros de sistemas embebidos están divididos en cuanto a si estos diseños son o no, sistemas embebidos, incluso aunque en la actualidad estos sistemas se discuten a menudo, como tal, entre estos mismos diseñadores.

Las definiciones tradicionales de sistemas embebidos deben seguir evolucionando, o un nuevo campo de los sistemas computacionales debe ser designado para incluir estos sistemas más complejos que últimamente están determinados por otros campos de la industria, por ahora, como no hay dicho campo para los sistemas computacionales que quedan entre los sistemas embebidos tradicionales y los sistemas computaciones de propósito general (PC), el autor Noergaard [2005] apoya el punto de vista evolutivo de los sistemas embebidos que incluye estos tipos de diseño de sistemas computacionales.

A continuación en la Tabla 2 se muestra una clasificación de los sistemas embebidos más comunes en sus tipos de mercado, este tipo de clasificación es muy utilizada por los fabricantes de circuitos electrónicos en la venta y asesoría de sus productos, ya que por lo general los elementos de cada grupo de mercadeo posee unas características de desempeño similares, lo cual permite especificar de mejor manera los servicios al cliente final. Se puede observar que debido a los tipos de aplicaciones, se hace difícil poder enmarcar todos estos dispositivos en una sola definición puntual que los involucre a todos.

Tabla 2. Ejemplos de sistemas embebidos y sus mercados

Mercado	Dispositivo embebido
Automotriz	Sistema de encendido
	Control del motor
	Sistema de frenos, i.e., sistema antibloqueo
Consumo electrónico	Televisores análogos y digitales
	Set-Top-Boxes (DVD, videograbadores, etc.)
	Asistente personal de datos (PDA)
	Aplicaciones de cocina (Nevera, Horno microondas, etc.)
	Consolas de juegos, juguetes
	Teléfonos, celulares
	Cámaras
	Sistemas de posicionamiento global (GPS)
Control industrial	Robótica y sistemas de control (Manufactura)
Medicina	Bombas de infusión
	Máquinas de diálisis
	Dispositivos de prótesis
	Monitores cardiacos
Redes de datos	Routers
	Switch
	Gateway
Automatización de oficinas	Máquinas de fax
	Fotocopiadoras
	Impresoras
	Monitores
	Escáner

Noergaard, Embedded Systems Architecture, 2005.

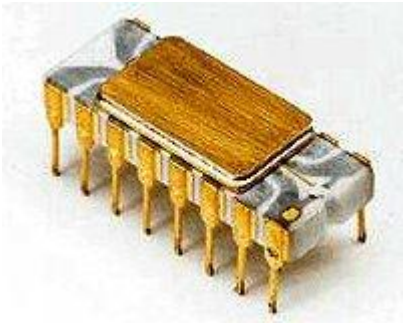
4.2. HISTORIA

La historia de los sistemas embebidos inicia con el microprocesador 4004 realizado por el fabricante de memoria Intel, anterior a este chip, el software sólo se utilizaba en aplicaciones stand-alone y ordenadores centrales, normalmente todos los dispositivos electrónicos eran diseñados completamente con hardware personalizado.

En 1969 el fabricante de calculadoras japonés *Busicom* encargó al fabricante de componentes electrónicos *Intel* un circuito integrado para su nueva línea de 12 calculadoras que diferían sólo en conjuntos de características. Pero Intel se dio cuenta de que si utilizaban los procesos de siempre para realizar ese circuito, gracias al gran costo de dicho proceso, las calculadoras saldrían tan caras como los miniordenadores de aquella época y no tendrían la capacidad de éstos, el ingeniero Ted Hoff de Intel propuso que era mejor hacer un chip programable que tuviera las funciones de una calculadora en vez de conectar dentro del chip los circuitos lógicos de la calculadora y así utilizar el mismo hardware de cada dispositivo, junto con el software específico del dispositivo reduciría los costes de desarrollo, gracias a que el costo de producción en masa es mucho más barato que el costo de producción de cada chip. Las calculadoras fueron un rotundo fracaso, pero el microprocesador 4004 fue un gran éxito [Etengabeko, 2010].

El 4004 mostrado en la Figura 1, fue el primer procesador de la época, tenía una arquitectura de Microprocesador de 4 bits y contenía 2.300 transistores, con el pasar del tiempo estos sistemas obtuvieron grandes adelantos de la noche a la mañana, y su uso aumentó de forma constante durante los siguientes años de esa década, actualmente los procesadores contienen alrededor de 230 millones de transistores.

Figura 1. Microprocesador 4004



Etengabeko, 2010.

Los primeros microprocesadores fueron incrustados en aplicaciones incluidas en sondas espaciales no tripuladas, luces de tráfico computarizadas, y el control de vuelo de aeronaves. En la década de 1980, los sistemas integrados se fueron metiendo silenciosamente en la ola de la edad de microcomputadoras y trajo microprocesadores a cada parte de nuestra vida personal y profesional. [Barr, 1999]

Parece inevitable que el número de sistemas embebidos siga aumentando rápidamente, el tamaño del mercado de sistemas embebidos pueden ser analizados desde una variedad de perspectivas. En cuanto a la cantidad de procesadores que se utilizan actualmente, se estima que alrededor del 79% de todos los procesadores se utilizan en sistemas integrados. Ya en el año 1996, se estimó que el estadounidense promedio entró en contacto con 60 microprocesadores por día, hoy en día los coches de gama alta contienen más de 100 procesadores, estas cifras son mucho más grande que lo que normalmente se espera, ya que la mayoría de las personas no se dan cuenta que están utilizando procesadores.

De los nueve millones de procesadores fabricados en 2005, menos del 2% se convirtieron en el cerebro de los nuevos PC, Mac y estaciones de trabajo Unix.

Los otros 8,8 mil millones fueron parte de sistemas embebidos de aplicaciones específicas. Son la esencia de todos los dispositivos electrónicos modernos, desde juguetes hasta controladores de plantas de energía nuclear, a partir de estos procesadores funcionan las fábricas, se hace la administración de sistemas de armamento y permiten el flujo mundial de información. [Netrino, 2006]

Los sistemas embebidos (a veces también llamados *empotrados*) son la base de la era llamada post-PC, en la que el manejo de la información se aleja cada vez más de los ordenadores tradicionales, siendo incrustada en sistemas portables y pequeños. El creciente número de aplicaciones resulta en la necesidad de tecnologías de diseño que apoyen el desarrollo de sistemas embebidos, las tecnologías y herramientas actualmente disponibles aún tienen limitaciones importantes, Por ejemplo, todavía hay una necesidad de mejores lenguajes de especificación, herramientas de generación de modelos, verificadores de intervalos de tiempo, sistemas operativos en tiempo real, técnicas de diseño de baja potencia y técnicas de diseño de sistemas confiables. [Marwedel, 2006]

4.3. TENDENCIAS

Según ha venido dándose la evolución de los sistemas embebidos, se han ido caracterizando ciertas tendencias en las metodologías de diseño y en los elementos que componen los nuevos sistemas embebidos que salen al mercado, a continuación se listan algunas:

Una tendencia muy importante en cuanto a hardware es que éste tenderá a ser abierto (con diseño e interfaces abiertos y modificables por el usuario), de forma que se pueda utilizar para todo tipo de aplicaciones (arquitecturas interoperables). Asimismo, también se generalizará el uso de sistemas operativos y software abierto para programas y aplicaciones que deban funcionar en tiempo real, permitiendo darle un correcto funcionamiento de acuerdo con los conceptos de

determinismo, sensibilidad, controlabilidad, fiabilidad y tolerancia a fallos. Como consecuencia de la mejora de la tolerancia a fallos (confiabilidad) se podrá trabajar en condiciones degradadas, alargando la vida útil de los sistemas con inteligencia embebida, el hecho que las arquitecturas puedan soportar cualquier modo de trabajo en estado degradado puede convertirse en la clave para el éxito comercial de muchos productos, ya que se asegura el funcionamiento de los mismos pese a las malas condiciones, Otro tema destacable de este apartado es el que hace referencia a la evolución de los chips ligada a su uso en Sistemas Embebidos. Se prevé que entre el 2015-2020 los chips tendrán integrados métodos de autoconfiguración y autodiagnóstico que les permitirá adaptarse de forma óptima a las tareas en cada situación y trabajar en estado degradado.

El continuo progreso en la tecnología microelectrónica está permitiendo que la tendencia actual sea la de integrar todos los componentes de un computador en un solo circuito. Es el llamado “System on Chip (SoC)”, que se prevé que se implante en todos los ámbitos en el período 2015-2020. Paralelamente, se conseguirá un despliegue creciente del “Network on Chip”, que se puede definir como un “System on Chip” con capacidades de comunicación integradas. Los Sistemas Embebidos con soluciones SoC implementadas presentan una serie de ventajas que podrían resumirse en: una elevada fiabilidad, tamaño reducido, mayor rendimiento, menor consumo energético y menor coste. [Opti, 2009]

La tendencia del diseño apunta cada vez más a que el hardware sea dependiente del software, la clave tecnología para futuros diseños es mejorar las cuestiones claves en la integración de las partes de un SoC, estas claves son la creación de una continuidad entre el software y el hardware integrado. Esto requiere una nueva tecnología llamada Hardware dependiente del Software (HDS) para integrar software embebido en las plataformas de hardware. El concepto HDS es una innovación prometedora enfocada a la integración de los SoC. Este concepto facilita:

- Diseño concurrente de hardware y software embebidos que conduce a una reducción del time-to-market.

- El diseño modular de partes de hardware y software, conduce a un mejor dominio de los sistemas complejos.

- Facilidad mundial de validación de SoC incluido hardware y software embebido que conduce a una mayor fiabilidad y una mejor calidad de servicios.

Aun cuando el concepto de interfaces clásicas de hardware lleva mucho tiempo, la migración a hardware dependiente de software de diseño y la aplicación de SoC trae nuevos retos que hay que explorar. [Ahmed, 2004]

En cuanto a las interfaces de comunicación entre estos sistemas y su entorno, cabe destacar que se desarrollarán interfaces Humano-Máquina adaptadas a cualquier uso para interactuar con equipos que contengan Sistemas Embebidos, asimismo, se prevé que en el período 2021-2025 las interfaces de los Sistemas Embebidos tengan elementos de traducción automática y de síntesis de voz totalmente fiables y fáciles de integrar. [Opti, 2009]

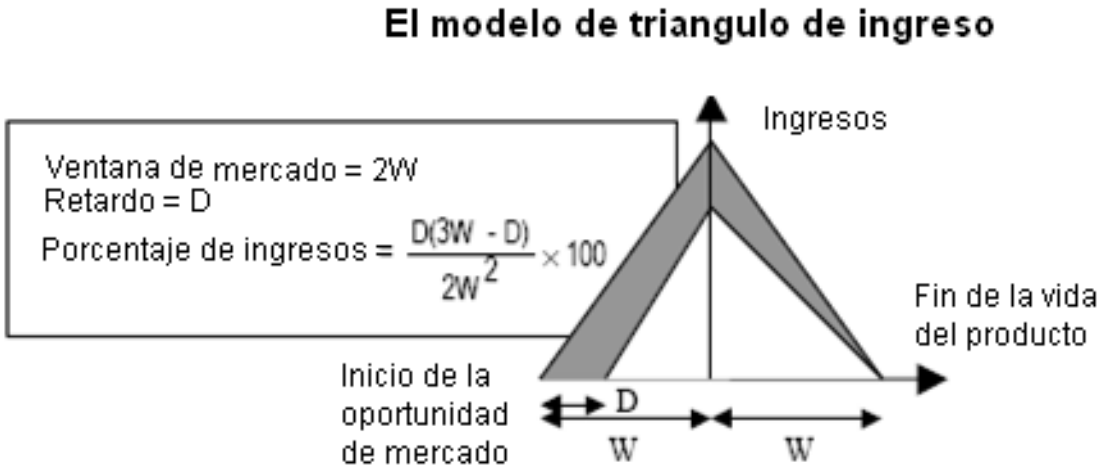
Debido a la globalización y la alta competitividad del mercado electrónico, hoy en día no solo se compite contra los mercados locales, sino también contra los globales, es necesario tener proceso de producción rápidos, eficientes y versátiles, puesto que toda empresa busca incrementar las ganancias de sus productos al máximo, no se pueden dar el lujo de generar demoras en sus procesos de desarrollo para sacar nuevos productos al mercado, es por esto que se demuestra que adoptando una metodología de diseño, se puede disminuir el time-to-market el cual es el tiempo que demora un producto desde que es concebido como idea, hasta que es lanzado al mercado, el modelo del triángulo de Ingresos en la Figura

2, muestra el efecto de una entrega tardía en el mercado, El modelo supone un pico en el medio de vida del producto y la pérdida de ingresos está representada en el área oscura, en este modelo los dispositivos programables no tienen el costo NRE (Non-Recurring Engineering, el termino de ingeniería no recurrente se refiere al costo de investigar, desarrollar, diseñar y probar un nuevo producto una sola vez), herramientas CAD más baratas y menos time-to-market, aumentan las ganancias sobre un producto, más tarde, al hacer la conversión a ASIC (Application-Specific Integrated Circuit, es un circuito integrado hecho a la medida para un uso en particular con base en un diseño del usuario) es posible reducir los costes de producción y la NRE asociados es menor que el diseño de un sólo ASIC. [Nunes, 2008]

Uno de los objetivos prioritarios para los ejecutivos de las empresas de bienes de consumo de producto, es acelerar la velocidad en que llevan nuevos productos al mercado, aunque siempre los procesos de empacado son los cuellos de botella que dificultan el cumplimiento de este objetivo, lo que hacen es replantear la forma en que convergen todas las actividades durante el desarrollo y empaquetado del producto, buscando del mismo modo la reducción de los costos de desarrollo y acelerar la consecución de ingresos. [Kodak, 2009]

Actualmente en la industria electrónica, la gran mayoría de desarrollos se hacen a través de prototipos virtuales, los cuales son modelos de software que emulan sistemas completos que proporcionan a los ingenieros de software, ambientes de desarrollo, en un tiempo mucho más rápido, que el que tardaría esperar hasta que el hardware esté listo, de este modo los prototipos virtuales permiten el desarrollo simultáneo de hardware y software, acortando significativamente, el tiempo de integración de software y la aceleración de sus productos al mercado. Debido a que se basan en modelos de software, los prototipos virtuales ofrecen una eficacia inigualable para el desarrollo y depuración de los diseños. [Synopsys, 2010]

Figura 2. Modelo de triangulo de ingreso



NUNES, C; Otros. Hardware-Software Codesign of Embedded Systems. 1998.

Como se mencionó anteriormente, hoy en día los sistemas embebidos están presentes en casi la totalidad de los sistemas electrónicos, debido a esto existe también una infinidad de productos que cada día están sometidos a procesos de investigación para crear nuevos productos más eficientes e innovadores, por este motivo no es correcto hablar de algún tipo de empresa u organización que lidere esta rama de investigación, aunque muchos aspectos de diseño de la electrónica están estandarizados en toda la industria, la metodología de diseño no es uno de ellos. De hecho, sería difícil incluir en una sola metodología normalizada todos los proyectos de electrónica, puesto que pueden existir diferencias significativas entre cada tipo de diseño en dos aplicaciones similares. Por otra parte, el conjunto de herramientas disponibles para los diseñadores evoluciona muy rápidamente. Por lo tanto, cada organización suele utilizar o definir una metodología propia basada en los tipos de los proyectos de diseño que realiza. [Ashenden, 2008]

En cuando a los niveles de abstracción del hardware para la elaboración del diseño, se espera que siga como es planteado en la Tabla 3. En esta tabla se combinan varios estudios. Las dos primeras columnas el resultado de una estudio de los niveles de abstracción utilizados en el hardware y el dominios de software, cada fila corresponde a una capa de abstracción específica. Estos son presentados por orden creciente de abstracción. Las primeras columnas dan el concepto abstracto de cada nivel de abstracción y un modelo típico utilizado en la literatura. La tercera columna describe las tecnologías necesarias para perfeccionar la capa de abstracción a la siguiente en la tabla. De acuerdo con el actual estado del arte y la dificultad de aplicar la tecnología, se intenta construir calendario a largo plazo (última columna). [Ahmed, 2004]

En el ámbito académico existen alrededor del mundo maestrías y doctorados sobre sistemas embebidos en los que cada vez se investiga más sobre las metodologías de diseño para sistemas embebidos, entre las universidades más conocidas que poseen áreas de investigación en el tema está el MIT (Massachusetts Institute of Technology) el cual posee un área de desarrollo en sistemas embebidos inteligentes y la universidad de california, Berkeley en la cual los sistemas embebidos han sido un área de investigación sólida y que quizás es la universidad líder en el área debido a su amplio y constante trabajo en las metodologías de diseño de sistemas embebidos, siendo la educación de sistemas embebidos relacionada íntimamente entre los programas de estudios con el programa de investigación. [Berkeley][Massachusetts][Sangiovanni-Vincentelli, 2005]

Tabla 3. Plan de trabajo de los niveles de abstracción y tecnologías clave.

Nivel Abstracción		**Modelos o lenguajes típicos**	**Claves tecnológicas**	**Año de marketing**
HARDWARE	**SOFTWARE**			
Retraso de compuerta	Todo explicito	RTL (Register Transfer Level)	Optimización lógica	1995
Envolvimiento de interfaces	Arquitectura local	TML (Transaction Model Level)	Diseño y optimización de interfaces HW - SW	2005 – 2007
Interconexión/ Sincronización		Nivel de transporte/ MPI (Message Passing Interface)	-Network-on-Chip -Interface de red -Configuración	2005 - 2010
Comunicación		SDL (Specification and Description Language)	Particionamiento de la comunicación computacional	2010 - 2015
Mapeo		BSP (Bulk-Synchronous Parallel),	Asignación de tareas, automatización programada	2015 - 2020
Descomposición		ParLog (PARallel LOGic)	Particionamiento automático	>>
Concurrencia		Haskel, PPP	generación de modelos	>>

Fundación OPTI, Tendencias y aplicaciones de los Estudio de Prospectiva Sistemas Embebidos en España, 2009.

4.4. OPORTUNIDADES Y DESAFÍOS

El objetivo final en el desarrollo de sistemas embebidos es el de diseñar el hardware y el software en todos los niveles de abstracción, tradicionalmente la investigación se ha concentrado en el particionamiento de hardware/software, pero sin resolver el problema de la abstracción de las plataformas de hardware. En lugar de utilizar modelos especiales de hardware, El próximo paso en la

automatización sería la síntesis de transferencia de datos, la sincronización y la interconexión, la comunicación, y particiones de hardware/software. [Jerraya, 2005]

Actualmente los sistemas de hardware están diseñados como la composición de interconexiones de componentes inherentemente paralelos. Los componentes individuales están representados por modelos analíticos (ecuaciones), que especifican sus funciones de transferencia. Estos modelos son deterministas (o probabilísticos), y sus composición se define mediante la especificación de cómo fluyen los datos a través de múltiples componentes. Los sistemas de software, por el contrario, están diseñados a partir de componentes secuenciales, como objetos y temas, cuya estructura cambia a menudo de forma dinámica (componentes se crean, eliminan y pueden migrar). Los componentes están representados por modelos computacionales (programas), cuya semántica se define operacionalmente por un motor de ejecución abstracto (también conocido como máquina virtual, o un autómata). Las máquinas abstractas puede ser no deterministas, y su composición se define especificando cómo controlar los flujos a través de múltiples componentes, por ejemplo, la acción atómica de los procesos de independencia pueden ser intercaladas, posiblemente limitado por un conjunto fijo de primitivas de sincronización.

Por lo tanto, el funcionamiento básico para la construcción de modelos de hardware es la composición de funciones de transferencia, la operación básica para la construcción de modelos de software es el producto de los autómatas. Estos son dos puntos de vista radicalmente diferentes para la construcción sistemas dinámicos a partir de componentes básicos: es decir, una analítica (basada en ecuaciones), la otra computacional (basado en sistemas informáticos).

El punto de vista analítico es frecuente en ingeniería eléctrica; el punto de vista computacional, en ciencias de la computación: la representación de un circuito en

netlist (descripción en texto de la conectividad de un diseño electrónico.) es un ejemplo de un modelo analítico, cualquier programa escrito en un lenguaje imperativo es un ejemplo de un modelo de cálculo. Dado que ambos tipos de modelos tienen fortalezas y debilidades muy diferentes, las implicaciones en el proceso de diseño son dramáticas. [Henzinger, 2006]

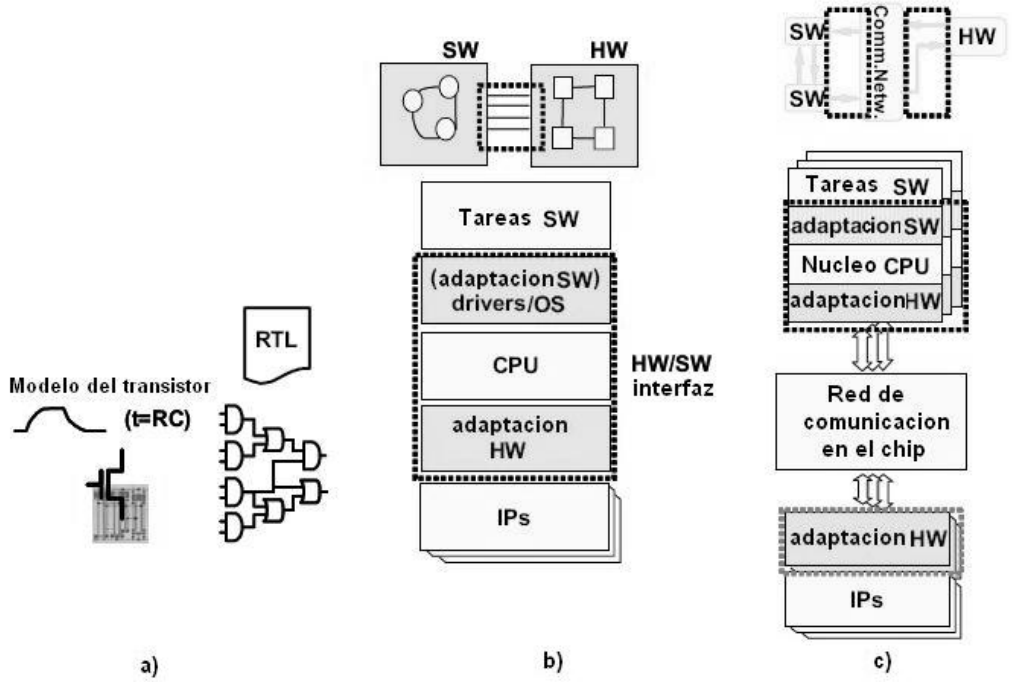
Las nuevas generaciones de diseños de software intensivo y que cuentan con ventanas de mercado a corto plazo y son cada vez con funcionalidades más complejas y fiables. Los enfoques actuales de diseño ASIC son altamente difíciles de escalar para tales SoC con varios procesadores funcionando en paralelo. El diseño de estos nuevos sistemas por medio de métodos clásicos da unos costes inaceptables para la realización y los retrasos. Los diseñadores tendrán un problema importante en la entrega de productos competitivos en el mercado que al mismo tiempo respeten un corto “time-to-market”, sean de bajo costo, y sean diseños complejos y fiables en SoC de varios procesadores. Los desafíos claramente están en el diseño de nuevas metodologías innovadoras.

La Figura 3, muestra la evolución de los niveles de abstracción en diseño de chips. Desde el diseño a nivel RTL, la abstracción se ha aplicado a los componentes de hardware e interconexiones (Figura 3 a), en cuanto a la interconexión, cables se abstraen de las líneas de metal a nivel de diseño para señales a nivel RTL. [Ahmed, 2004]

Por encima de RTL, la abstracción tiene que ser aplicada a software y componentes de hardware (Figura 3 b). Puesto que los componentes de software se ejecutan en los procesadores, la abstracción de la interconexión entre el software y componentes de hardware es totalmente diferente de la actual abstracción de los cables entre los componentes de hardware. De hecho, los componentes de software no se comunican con el mundo externo a través de cables, sino a través de llamadas a funciones (dispositivo piloto) o las

instrucciones de montaje (load/store). La interfaz hardware/software tiene que manejar dos interfaces diferentes: uno sobre el lado del software usando la API (Application Perihelia Interface es el conjunto de funciones utilizado como una capa de abstracción) y una en el lado del hardware utilizando alambres.

Figura 3. Evolución de los niveles de abstracción en el diseño de SE



Ahmed, Long Term Trends for Embedded System Design, 2004.

Esta heterogeneidad hace que el diseño de la interfaz hardware/software sea muy difícil y lleva mucho tiempo debido a que el diseño requiere el conocimiento de los programas y el hardware y su interacción. Por lo tanto, un nuevo tipo de diseñador, es decir, el diseño de la integración de hardware/software sea necesario. En términos de madurez de la técnica de diseño, el diseño de la interfaz de hardware/software sigue siendo uno de los cuellos de botella en el diseño SoC y, por tanto, tiene que ser dominado. [Ahmed, 2004]

Los enfoques actuales para el problema de codiseño de hardware-software se quedan cortos, se cree en una de dos razones. El modelo formal que no es definido lo suficientemente abstracto para ser utilizado como una implementación independiente de la representación en los procesos de diseño del sistema, De lo contrario, el modelo no es suficientemente detallado para la síntesis eficiente como una mezcla de componentes de hardware y software. [Chiodo, 1994]

5. DISEÑO DE SISTEMAS EMBEBIDOS

En esta sección se hace una recopilación de las principales metodologías de diseño que existen para el desarrollo de sistemas embebidos, se inicia mostrando algunas generalidades sobre los diseños de sistemas en general, posteriormente se describen los modelos de ciclo de vida de los productos que son los patrones o flujos de etapas a desarrollar, luego se hace énfasis sobre las metodologías existentes con la descripción de sus etapas, por último se muestra una tabla comparativa y se propone un modelo de ciclo de vida aplicado al desarrollo de un sistema de medición de datos, a partir del cual se pretende evaluar el protocolo a diseñar.

5.1. PROCESO DE DESARROLLO DE NUEVOS PRODUCTOS

De manera general el desarrollo de un sistema embebido que se pretende introducir al mercado cumple las mismas etapas de diseño de un producto de consumo común, Desde el punto de vista de la ingeniería de productos, independientemente de cuál sea el enfoque organizacional que se utilice para el desarrollo de nuevos productos, los pasos que se siguen son casi siempre los mostrados en la Figura 4.

- **Generación de la idea**: Las ideas se pueden generar a partir del mercado o de la tecnología. Las ideas del mercado se derivan de las necesidades del consumidor. La identificación de las necesidades del mercado puede llevar entonces al desarrollo de nuevas tecnologías y productos para satisfacer estas necesidades, por otro lado las ideas también pueden surgir de la tecnología disponible o nueva.

- **Selección del producto**: No todas las ideas nuevas deben desarrollarse para convertirlas en nuevos productos. Las ideas para nuevos productos

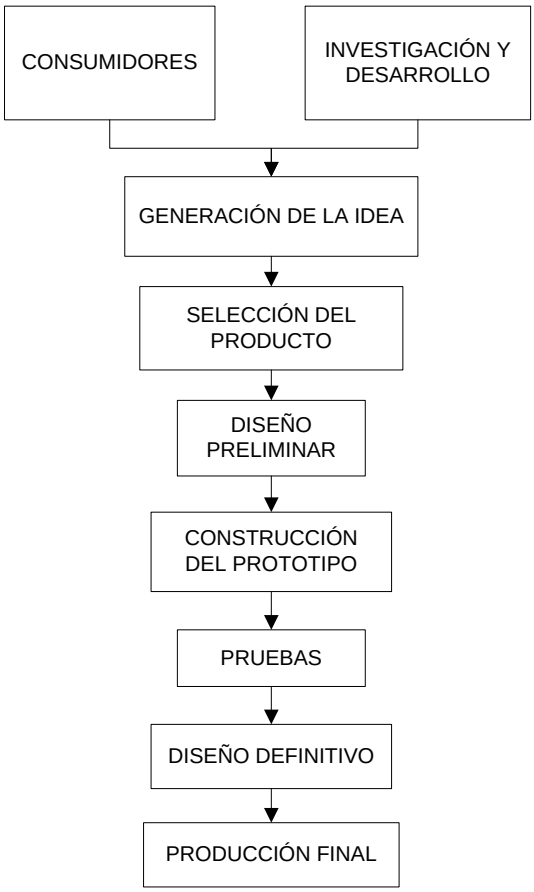
deben pasar por lo menos tres pruebas: el potencial del mercado, factibilidad financiera, compatibilidad con operaciones. Antes de colocar la idea de un nuevo producto en el diseño preliminar se le debe someter a los análisis necesarios que se organizan alrededor de estas tres pruebas. El propósito del análisis de selección es identificar cuáles son las mejores ideas.

- **Diseño preliminar:** Esta etapa del diseño de un producto se relaciona con el desarrollo del mejor diseño para la idea del nuevo producto. Cuando se aprueba un diseño preliminar, se puede construir un prototipo para someterlo a pruebas adicionales y análisis. En el diseño preliminar se toma en cuenta: costo, calidad y rendimiento del producto. El resultado debe ser un diseño de producto que resulte competitivo en el mercado y que pueda producirse.

• Construcción del prototipo: La construcción del prototipo puede tener varias formas diferentes, se pueden fabricar a mano varios prototipos que se parezcan al producto final.

• Pruebas: Las pruebas en los prototipos buscan verificar el desempeño técnico y comercial. Una manera de apreciarlo es construir suficientes prototipos como para apoyar una prueba de mercado. Las pruebas de mercado casi siempre duran entre seis meses y dos años y se limitan a una región geográfica pequeña. El propósito de una prueba de mercado es obtener datos cuantitativos sobre la aceptación que tiene el producto entre los consumidores.

Figura 4. Proceso de desarrollo de nuevos productos



Díaz, Diseño de producto, 2009.

- **Diseño definitivo del producto:** Como resultado de las pruebas en los prototipos se pueden incorporar ciertos cambios en el diseño definitivo. Cuando se hacen cambios, el producto puede someterse a pruebas adicionales para asegurar el desempeño del producto final. La atención se coloca en la terminación de las especificaciones de diseño para que se pueda proceder con la producción.

- **Producción final:** Como etapa final se diseñan los procesos de producción en masa teniendo en cuenta la disminución de costos que se puede obtener en la economía a escala, así como buscar los mejores proveedores y canales de distribución. [Díaz, 2009]

A todo este conjunto de pasos para obtener un producto se llama método de diseño, este término se refiriere al proceso sistemático de diseño, verificación y la preparación para la fabricación de un producto. Un método de diseño especifica las tareas emprendidas, la información requerida y producida por cada tarea, las relaciones entre las tareas, incluyendo las dependencias y secuencias, así como las herramientas CAD utilizadas. Una metodología de diseño madura especifica las medidas que se harán del proceso de diseño, tales como la adhesión al programa y presupuesto, y el número de errores de diseño detectados. Los datos acumulados de los proyectos anteriores pueden utilizarse para mejorar el método de diseño para proyectos posteriores. El beneficio de un buen método es que hace que el proceso de diseño sea más fiable y predecible, lo que reduce riesgos y costes, incluso en los proyectos de pequeños diseños también hay beneficios, así sean de menor escala. [Ashenden, 2008]

5.2. MODELOS DE CICLO DE VIDA

El desarrollo de un sistema embebido es un proceso metódico en el que siguen varias etapas siendo todo ese desarrollo un proceso completo que va desde la planificación del producto a desarrollar, hasta obtener un artículo final, Dentro de las áreas de la ingeniería electrónica y de sistemas se encuentra *el ciclo de vida del desarrollo de un producto (SDLC: System Development Life Cycle*), el cual es el método de diseño más aplicado para el desarrollo de sistemas de información contemporáneos [Valachi, 2006], Un modelo de ciclo de vida es una vista de las actividades que ocurren durante el desarrollo de un producto, intenta determinar el

orden de las etapas involucradas y los criterios de transición asociadas entre estas etapas.

Un modelo de ciclo de vida:

- Describe las fases principales de desarrollo.
- Define las fases primarias esperadas de ser ejecutadas durante esas fases.
- Ayuda a administrar el progreso del desarrollo.
- Provee un espacio de trabajo para la definición de un detallado proceso de desarrollo.

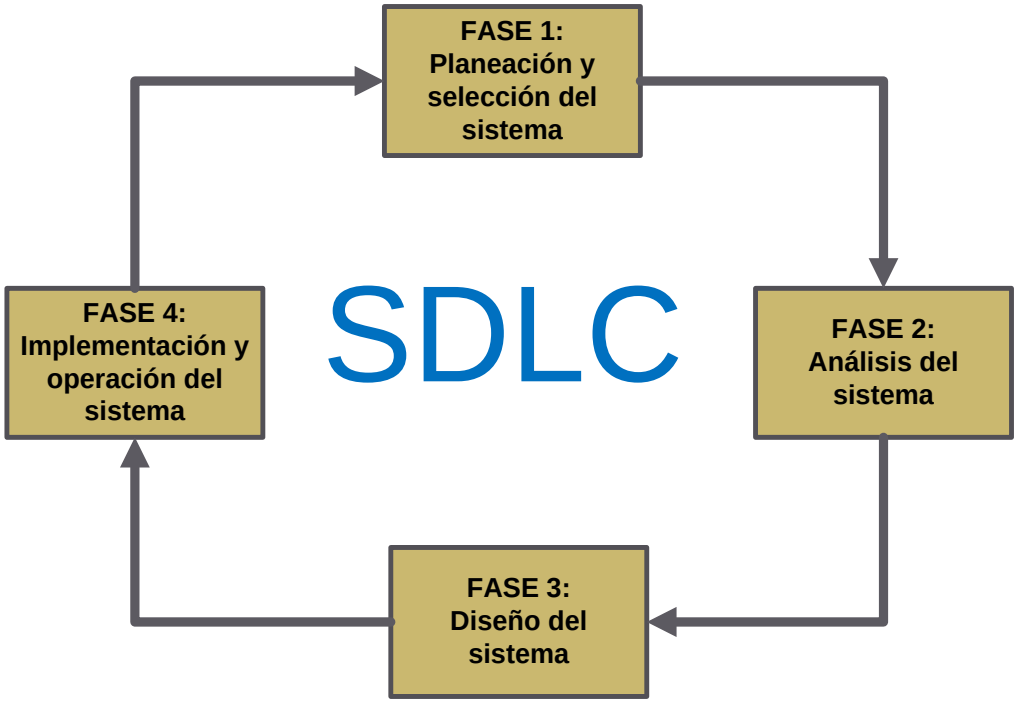
Así, los modelos por una parte suministran una guía para los ingenieros con el fin de ordenar las diversas actividades técnicas en el proyecto, por otra parte suministran un marco para la administración del desarrollo y el mantenimiento, en el sentido en que permiten estimar recursos, definir puntos de control intermedios, monitorear el avance, etc. [Cantone, 2008]

Las cuatro claves de este método son:

- Planeación y selección
- Análisis
- Diseño
- Implementación y operación

A continuación se describen los modelos básicos de ciclo de vida, en cada uno se podrá observar las 4 etapas del SDLC, se observa también que la evolución de los modelos tienden a una mayor flexibilidad entre las etapas para poder hacer modificaciones en una etapa anterior una vez ya se esté trabajando en otra y que eso a su vez no incremente los costos ni el normal desarrollo del producto,

Figura 5. Ciclo de vida de diseño de sistemas



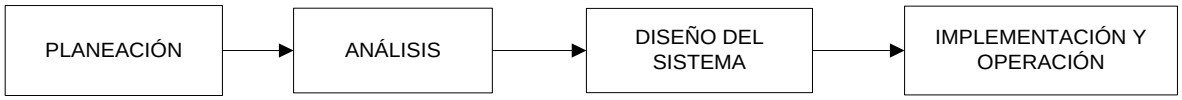
Valachi, Essentials of systems analysis and design, 2006.

Estos modelos son ampliamente utilizados en el campo de la ingeniería de software, rama de la ingeniería en la cual se ha ido mejorando y evolucionando a través del tiempo, pero de igual forma, gracias al tratamiento general sobre el desarrollo de un producto como tal, estos modelos son totalmente aplicables al desarrollo de sistemas embebidos.

5.2.1. Modelo de ciclo de vida lineal:

Es el más sencillo de todos los modelos ya que divide el proyecto en distintas partes para su elaboración, es importante que las partes en las que se divide el proyecto no tengan relación una de otras, este modelo es presentado en la Figura 6.

Figura 6. Modelo de ciclo de vida lineal



Ventajas:

- Las tareas se hacen secuenciales.
- Se puede reducir al mínimo los errores en la codificación.
- Tiene una planificación sencilla.

Desventajas:

- No se recomienda para proyectos que requieran de retroalimentación.
- En caso de encontrar un fallo es muy costoso remontar cualquiera de las etapas.

5.2.2. Modelo de ciclo de vida en cascada

La necesidad de conocer los requerimientos al principio del proyecto es primordial, este modelo es presentado en la Figura 7, se diferencia del modelo de ciclo de vida lineal en que cada etapa esta realimentada de la anterior lo cual brinda una flexibilidad mayor a la hora de corregir errores o detalles que pueden ser omitidos en una etapa.

Ventajas:

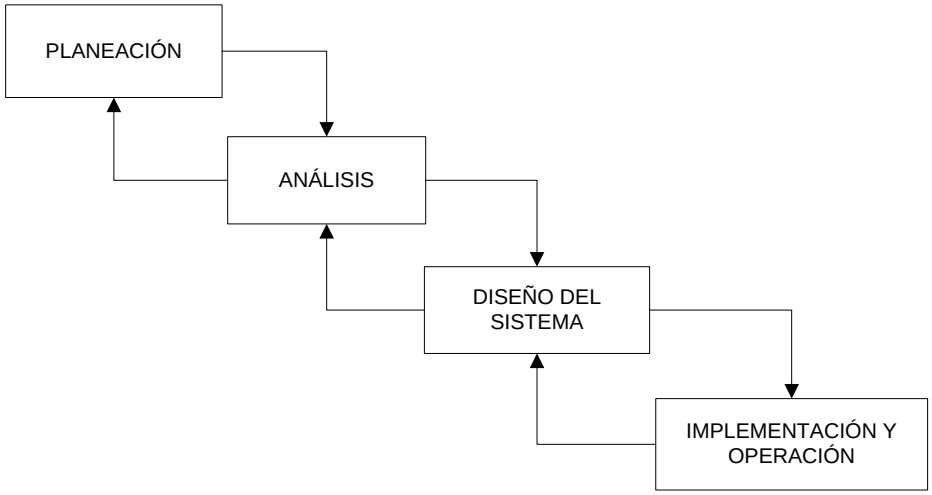
- Tiene una planificación sencilla.
- Produce un producto con un alto grado de calidad en el producto obtenido.
- Permite la corrección temprana de errores.

Desventajas:

- Se puede contar como un inconveniente la necesidad de conocer todos los requerimientos desde un principio.

- Si se comete algún error y no se detecta en la siguiente etapa es muy difícil volver atrás posteriormente. Cualquier error no detectado nos trae retrasos y un aumento en los costos.
- Los resultados no se ven hasta las etapas finales.

Figura 7. Modelo de ciclo en cascada puro

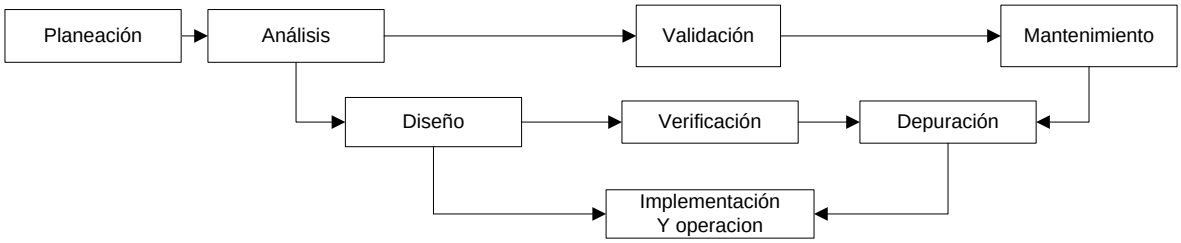


5.2.3. Modelo de ciclo de vida en V

Diseñado por Alan Davis, Contiene las mismas etapas que el anterior, más dos sub-etapas de retroalimentación, este modelo ofrece mayor garantía de corrección al terminar el proyecto, este modelo es presentado en la Figura 8.

Las ventajas y desventajas de este modelo son las mismas del ciclo anterior, con el agregado de los controles cruzados entre etapas para lograr una mayor corrección, se puede utilizar este modelo de ciclo de vida en aplicaciones, que si bien son simples, necesitan una confiabilidad muy alta[Cantone, 2008].

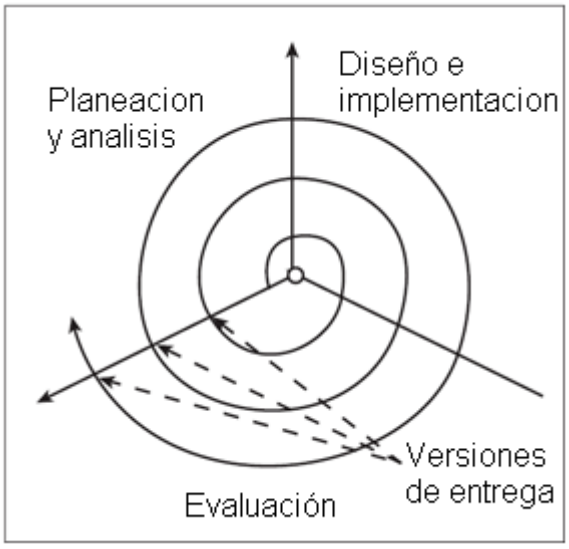
Figura 8. Modelo de ciclo V



5.2.4. Modelo de ciclo de vida evolutivo

Este modelo (Figura 9) acepta que los requerimientos del usuario pueden cambiar en cualquier momento, la práctica demuestra que obtener todos los requerimientos al comienzo del proyecto es extremadamente difícil, no solo por la dificultad del usuario de transmitir su idea, sino porque estos requerimientos evolucionan durante el desarrollo y de esta manera, surgen nuevos requerimientos a cumplir, el modelo de ciclo de vida evolutivo afronta este problema mediante una iteración del ciclo, luego de cada ciclo se obtiene una versión del producto.

Figura 9. Modelo de ciclo de vida evolutivo



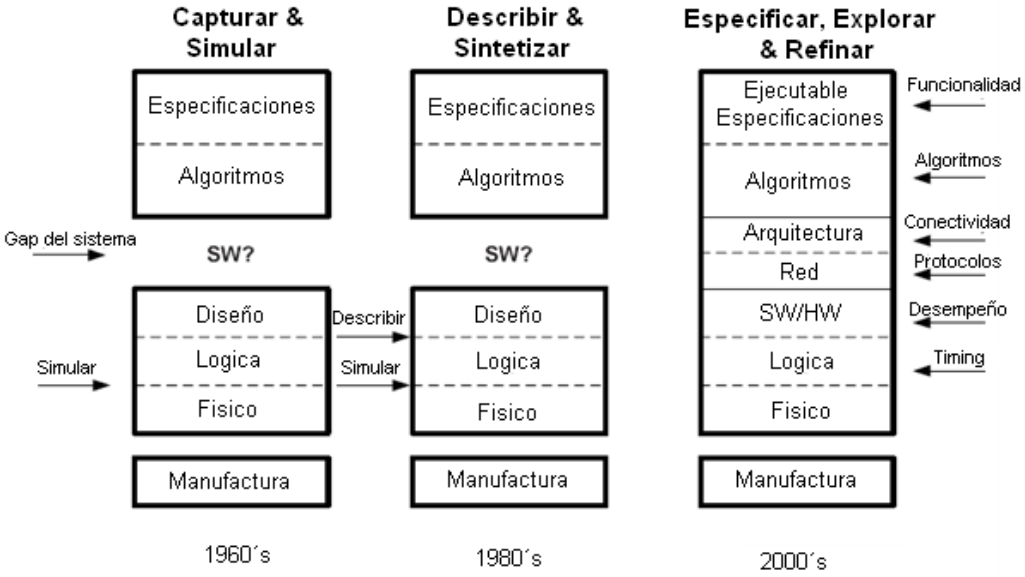
A partir de estos modelos de ciclo de vida, se generan unos flujos de diseños llamados metodologías, las cuales proponen una serie de etapas a seguir como método de planificación de los diseños a desarrollar.

5.3. METODOLOGÍAS DE DISEÑO

De los anteriores modelos se observan que dependiendo de la aplicación o de la forma en que el cliente desea el producto se puede aplicar el tipo más apropiado, ya sea que desde un principio se obtengan todas las especificaciones completas o si se desea entregar una serie de módulos al cliente, los cuales se desarrollan paso a paso, es muy común que cuando los diseñadores comienzan a diseñar e implementar soluciones que no han sido completamente especificadas conduce a la construcción de sistemas que no satisfacen las necesidades de los clientes y que incurren en el aumento de los costos y en el incumplimiento de los plazos establecidos, en el campo de los sistemas embebidos dicha problemática general de los sistemas de información no solo se conserva sino que se agrava hasta el punto que el 60% de las componentes que integran hardware y software deben ser rediseñadas luego de haber sido programadas [Ganssle, 1999; Gonzales 2008]. La problemática expuesta para los sistemas en general tiene mayor impacto en el caso de los sistemas embebidos.

El flujo de diseño ha ido cambiando con el aumento de la complejidad de los sistemas embebidos. Es posible indicar varios períodos que se tradujo en drásticos cambios en el flujo de diseño, herramientas y metodología, como se muestra en la Figura 10.

Figura 10. Evolución del flujo de diseño en los últimos 50 años



Gajski, Embedded System Design Modeling, 2009.

Metodología de captura y Simulación (1960 a 1980): En esta metodología, el diseño de software y hardware son separados por un sistema llamada brecha (gap), los diseñadores de software prueban algunos de los algoritmos y, en ocasiones, escriben el documento de requisitos y especificaciones inicial. Estas especificaciones son dadas a los diseñadores de hardware, que inician el diseño del sistema a partir del diagrama de bloques, sin embargo ellos no saben si su diseño cumple con las especificaciones de diseño hasta que el nivel de compuertas se ha producido. Cuando la red de compuertas ha sido capturada y simulada, los diseñadores podrían determinar si el sistema funcionó tal como se especifica. Por lo general, no en todos los casos, cuando el pliego de condiciones se modifica, cambian las capacidades de ejecución necesarias, este enfoque comenzó el mito que la especificación nunca está completa, tomó muchos años para que los diseñadores se dieran cuenta de que una especificación es independiente de su aplicación, lo que significa que la especificación puede estar siempre actualizándose, al igual que su puesta en práctica.

El principal obstáculo para cerrar la brecha entre el sistema de software y hardware, y por lo tanto entre la especificación y ejecución, fue el flujo de diseño en la que los diseñadores esperan a que el diseño de los niveles de compuerta esté terminado antes de la verificación de la especificación del sistema. En tal flujo de diseño son demasiados niveles de abstracción entre especificaciones del sistema y el nivel de diseño de compuertas para que los diseñadores de software participen.

Como los diseñadores capturan la descripción del diseño hasta el ciclo de diseño final sólo con fines de simulación, esta metodología se llama captura y simulación. Hay que tener en cuenta que no había documentación verificable antes de la captura del diseño de nivel de compuertas, ya que la mayoría de las decisiones de diseño se almacenaron de manera informal en la mente de los diseñadores.

Metodología de Descripción y Síntesis (finales de 1980 a finales de 1990): La década de 1980 trajo herramientas para la síntesis de la lógica que tienen un flujo de diseño significativamente modificado, ya que tanto el comportamiento y la estructura de un diseño fueron capturados en el nivel lógico. El diseñador especifica en primer lugar lo que quería en ecuaciones booleanas o descripciones FSM (Finite-State Machine o máquina de estados finitos, método matemático basado en estados según las entradas al sistema), a continuación, la síntesis herramientas generadas a la aplicación en términos de una netlists (netlists es un método de expresar circuitos electrónicos en texto plano) de nivel lógico. En esta metodología por lo tanto, el comportamiento o la función es lo primero, y la estructura o la aplicación vienen después. Por otra parte, ambas descripciones son simulables, lo cual es una mejora con respecto a la metodología de captura y simulación, porque permite no solo una verificación mucho más eficiente, sino que hace posible verificar las descripciones de equivalencia ya que tanto en las descripciones, en principio, puede reducirse a una forma canónica. Sin embargo,

los diseños de hoy son demasiado grandes para este tipo de control de la equivalencia.

A finales de 1990, el nivel lógico se había abstraído del Nivel de transferencia de registro (RTL) con la introducción de los modelos del ciclo con precisión y síntesis. Por lo tanto, ahora se tienen dos niveles de abstracción (RTL y niveles lógicos) y dos modelos diferentes en cada nivel (conductuales y estructurales). Sin embargo, la brecha del sistema aún persiste porque no existía relación entre RTL y sistemas de nivel superior.

Especificar, exploración y refinamiento de la metodología (principios de 2000 hasta la actualidad): Para cerrar el gap, se debe aumentar el nivel de abstracción de la RTL al nivel del sistema (SL) y se introdujo una metodología que incluye tanto software como hardware. En el nivel de sistema, es posible comenzar con un archivo ejecutable de especificación que representa el comportamiento del sistema, entonces se puede extender la metodología al nivel de sistema que incluya varios modelos con diferentes detalles que corresponden a decisiones de diseño diferentes. Cada modelo se utiliza para probar alguna propiedad del sistema: la funcionalidad, la aplicación de algoritmos, la conectividad, la comunicación, la sincronización, la coherencia, la expedición, el rendimiento o algunas métricas de diseño tales como rendimiento, potencia, etc. Así que se debe hacer frente a varios modelos con el fin de verificar el impacto de las decisiones en cada criterio de diseño de una especificación ejecutable hasta el RTL y en relación con el diseño físico. Se puede considerar cada modelo como una especificación para el modelo del siguiente nivel, en el que una aplicación más detallada se añade después de que se toman más decisiones de diseño. Es posible llamar a esto como la metodología Specify-Explore-Refine (SER), ya que consiste en una secuencia de modelos en los que cada modelo es un refinamiento de la anterior.

Así, la metodología SER sigue el proceso de diseño natural en el que los diseñadores especifican el primer intento, a continuación, estudian las posibilidades, y perfeccionan finalmente el modelo de acuerdo con sus decisiones. El flujo de SER por lo tanto puede ser visto como varias iteraciones de la metodología básica y Descripción y síntesis [Gajski, 2009].

Para los grandes proyectos del mundo real, después de haber una metodología claramente especificada, el apegarse a ella es indispensable. Muchos proyectos han fracasado, no por problemas técnicos, pero si por falta de control del proceso de diseño.

La mayoría de las metodologías de diseño usadas para los sistemas embebidos industriales han sido prestadas de las prácticas de diseño de software estándar, en donde el énfasis es puesto en los procesos de desarrollo más que en el contenido del diseño, sin embargo en los sectores de la industria donde la seguridad es una preocupación primaria, ha habido cambios hacia el uso de lenguajes que tengan garantías intrínsecas correctas y para que el análisis y la síntesis de métodos de gran alcance sean posibles. Irónicamente, mientras que en el caso de los sistemas de diseño de hardware el estado del arte intenta ir a lenguajes de adaptación como C y C++ típicamente usados para el software, el más avanzado paradigma en el diseño de software ha sido prestado del diseño de hardware, en particular el enfoque más elegante es la extensión del paradigma de diseño sincrónico al diseño de software. [Sangiovanni-Vincentelli, 2007]

Si se realiza una preparación adecuada en la definición del sistema, en la arquitectura de los oficiales administrativos, en la determinación de necesidades y en la comprensión de los riesgos, entonces las otras fases de desarrollo, pruebas y mantenimiento del dispositivo, será más sencillo, más rápido y más barato. Esto, por supuesto asume que los ingenieros responsables poseen las competencias necesarias. [Noergaard, 2005]

A continuación se exponen varias metodologías propuestas en la literatura de sistemas embebidos, puesto que en el momento no existe un estándar a seguir para el desarrollo de sistemas embebidos se busca reunir las características de estos métodos y los tipos de productos a los cuales va enfocada su aplicación, cada una de estas metodologías sigue de proceso o flujo de ejecución que se ajusta a uno de modelos de ciclo de vida, ya se lineal, en cascada, cíclico o evolutivo, etc. lo cual los hace a cada uno más propenso a algún tipo de aplicación, posterior al desarrollo de estas metodologías, se incluye una tabla resumen con las ventajas, desventajas y aplicaciones más comunes de cada metodología.

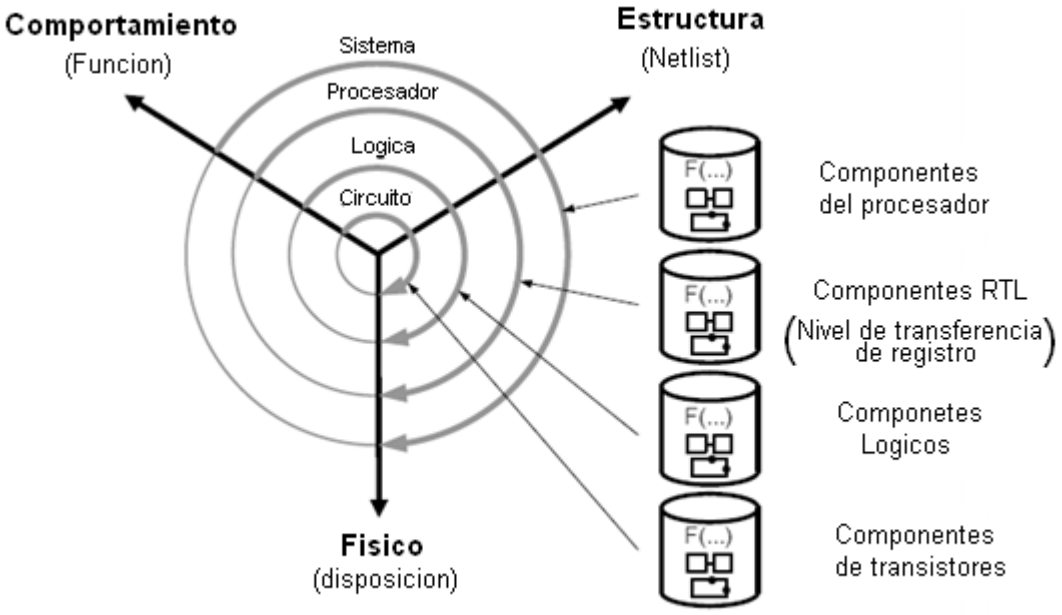
5.3.1. Metodología de abajo hacia arriba (Bottom-Up)

La metodología de abajo hacia arriba comenzó incluso antes de que se inventaran las herramientas CAD. Es todavía hoy usada en gran parte de la industria, al menos parcialmente, ya que sigue una metodología intuitiva de la construcción de las piezas antes de armar el producto entero. En una metodología típica de abajo hacia arriba, los diseñadores desarrollan componentes y, a continuación los almacenan en una biblioteca para el uso en el nivel de abstracción inmediatamente superior, como se observa en la Figura 11. El inicio de esta metodología parte del centro con la implementación de los circuitos simples basados en los transistores, a partir de los cuales se implementan unos componentes lógicos, y así sucesivamente hasta llegar al diseño completo de todo el sistema, partiendo de lo más básico, hasta el sistema completo como tal.

La ventaja de la metodología de abajo hacia arriba es que los niveles de abstracción son claramente separados, cada uno con su propia biblioteca. Esto permite que sea posible localizar mundialmente el diseño en cada nivel de abstracción, y facilitar así la gestión del diseño en cada nivel, puesto que cada grupo suministra una biblioteca de componentes para el siguiente nivel de

abstracción. La desventaja de este enfoque, sin embargo, es que las bibliotecas deben incluir todos los componentes posibles con todos los parámetros y que éstos deben ser optimizados para las Mediciones requeridas por el presente y posibles aplicaciones futuras. Esto es una tarea ardua e interminable puesto que es difícil anticipar las necesidades de abstracción en el nivel más bajo desde los niveles de abstracción más altos. [Gajski, 2009]

Figura 11. Metodología Bottom-Up



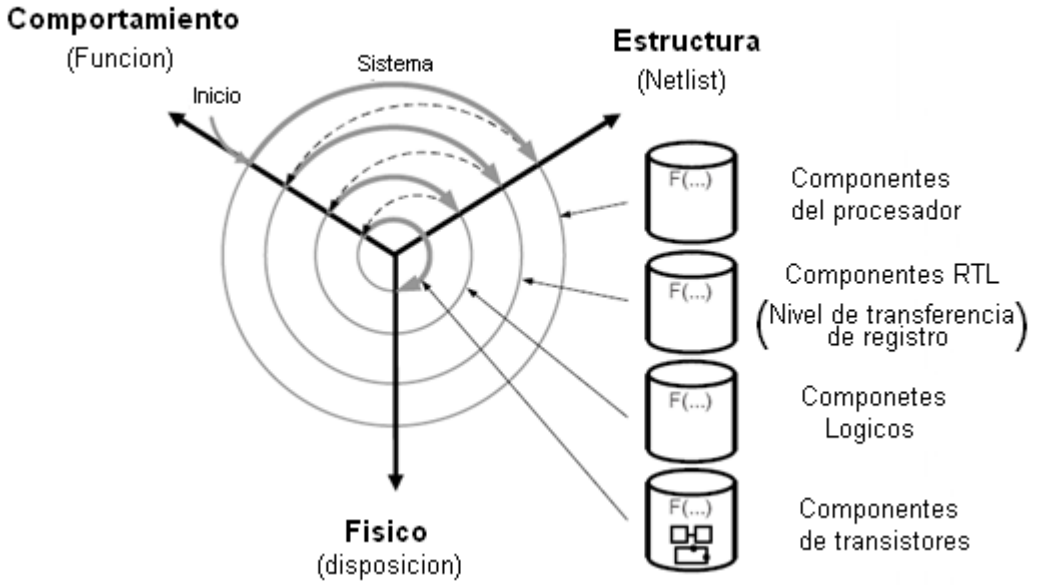
Gajski, Embedded System Design Modeling, 2009.

5.3.2. Metodología de arriba hacia abajo (Top-Down)

En contraste con la metodología de abajo hacia arriba, la metodología de arriba hacia abajo no trabaja un componente o capa del sistema hasta que todo el diseño está terminado. La metodología de arriba hacia abajo comienza con un particular Modelo de Computación (model of computation, MoC) a partir del cual, cada componente de una plataforma de sistema o estructura del sistema, tiene sus parámetros y valores de métrica necesarios, pero no su estructura o diseño. En el

siguiente nivel de abstracción cada Elemento de procesamiento (processing element, PE) o Elemento de Comunicación (communication elements CEs) es descompuesto en cada vez más en pequeños componentes RTL, de este modo al inicio del proyecto se formula un resumen del sistema, sin especificar detalles. Cada parte del sistema se refina diseñándolo con mayor detalle. Cada parte nueva es entonces redefinida, cada vez con mayor detalle, hasta que la especificación completa es lo suficientemente detallada para validar el modelo, la estructura del diseño de un sistema embebido requiere entonces la habilidad de hacer el diseño y la división del sistema desde lo general a lo específico mientras la implementación del sistema va de abajo hacia arriba.

Figura 12. Metodología Top-Down



Gajski, Embedded System Design Modeling, 2009.

En la Figura 12. Se observa que esta metodología inicia en la parte exterior del diagrama, partiendo desde el análisis del comportamiento general de todo el sistema y se sigue moviendo a través de la estructura y el comportamiento, para

dejar como último paso la disposición física de los circuitos del sistema. [Gajski, 2009]

Siguiendo la filosofía del diseño Top-Down, este método propone a partir de las descripciones generadas en el documento de requerimientos, ir diseñando poco a poco hasta llegar al nivel de detalle cada una de las 4 etapas propuestas, a medida que se realizan esta tareas se deben ir generado unos documentos guías llamados documentos de diseño donde se consigna toda la información del desarrollo y del producto final, estos documentos están compuestos por:

- **Documento de requisitos**: debe estar actualizado con toda la información actualizada del sistema, incluyendo las funciones requeridas para la operación, comunicaciones, requerimientos de almacenamiento, información de tiempos, e información de prioridades, también debe estar incluida información detallada respecto a la interfaz del usuario y finalmente, toda la información disponible en errores potenciales del sistema, métodos usados para identificar las condiciones de error y métodos para recuperarse de los errores.

- **La información tomada del documento de requerimientos**: Debe incluir información relativa a los siguientes:

    - Información de las tareas: se debe incluir una lista de todas las funciones necesarias que el diseño debe llevar a cabo.
    - Información de las comunicaciones: incluye toda la información sobre el tamaño y tipo de datos para las comunicaciones internas entre las funciones, las comunicaciones con los recursos fuera del sistema y cualquier almacenamiento temporal significativo.
    - Información de tiempo: Esto incluye no sólo los requerimientos temporales para las tareas individuales, sino también el calendario

general del sistema, incluyendo tanto el caso como el evento y el tiempo de respuesta.

  - Prioridad de la información: Esto incluye una descripción detallada de todos los modos del sistema y los eventos de activación que cambian el modo de operación del sistema.

- **Documentación sobre la fase de definición de las tareas del diseño a nivel de sistema**: Este debe incluir los nombres descriptivos para las diversas tareas en el sistema, las funciones del software que se han agrupado y las razones de combinación o exclusión de las funciones de varios programas.

- **Documentación sobre el plan de comunicaciones para el diseño**: Este debe incluir todas las revisiones del sistema, diagramas de flujo de datos, la lista preliminar de variables y toda la documentación relativa a las tareas relacionadas con el protocolo, las necesidades de memoria y la información puntual.

- **Documentación sobre el análisis temporal para el sistema**: Esto debería incluir todos los cálculos generados para determinar los pulsos del sistema, incluyendo tanto los casos de requisitos de tiempo óptimos, como los peores.

- **Documentación sobre las prioridades de los sistemas:** Incluye la lista actualizada de prioridad, utilizando el nombre de la tarea generada en la fase de definición de la tarea del diseño. Teniendo en cuenta cualquier tarea de menor prioridad que combinen funciones prioritarias y superiores.

- **Documentación sobre el sistema de detección de errores y recuperación en el diseño**: En particular, las condiciones de error

resultante de las tareas de la máquina de estados, posibles problemas de comunicación, y las posibilidades de daños en los datos generales.

[Ganssle, 2008, Arnold, 2000]

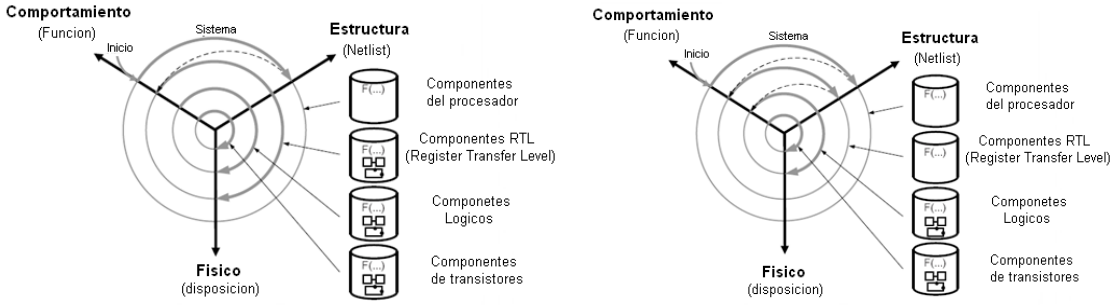
5.3.3. Metodología encuentro a medio camino (Meet-in-the-middle)

La mayoría de los diseñadores hoy en día usan algún tipo de metodología de “Meet-in-the-middle", a fin de aprovechar los beneficios tanto de las metodologías de abajo hacia arriba y arriba hacia abajo, al tiempo que minimizan sus inconvenientes. Es conveniente porque los estándares de diseño y herramientas de CAD en los niveles inferiores de las abstracciones son bien conocidas y desarrolladas, pero en el procesador y nivel de sistema no lo son. Si bien hay algunas herramientas en el nivel de procesador, casi ninguno, con excepción de las herramientas de simulación general, existe en el nivel de sistema. Una metodología “Meet-in-the-middle” permite que un diseñador pueda aprovechar de las herramientas disponibles para abstracciones de nivel inferior al tiempo que reduce el diseño en los niveles de mayor abstracción.

En general, en una metodología “meet-in-the-middle” se aplica una metodología de arriba hacia abajo en los niveles de abstracción más alto y una metodología de abajo hacia arriba para reducir la abstracción de los niveles mayores. [Gajski, 2009]

En la Figura 13, se observa que la metodología ofrece flexibilidad en el encuentro de los desarrollos de abajo hacia arriba y de arriba hacia abajo, según la aplicación que se esté desarrollando, en el caso de la imagen a la izquierda se observa que el encuentro se da en el nivel de componentes RTL, mientras que en la figura del lado derecho, el encuentro se da en el nivel de componentes lógicos.

Figura 13. Metodología “Meet in the middle”



Gajski, Daniel D. Embedded System Design Modeling, 2009.

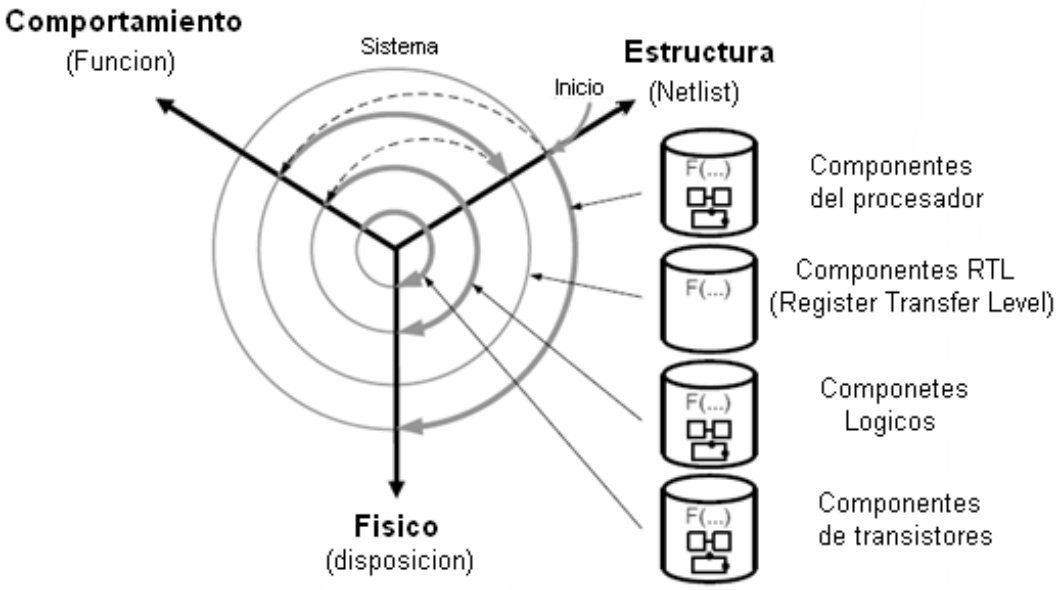
5.3.4. Metodología de plataforma

Las tres metodologías de diseño anteriores representan casos ideales de tres propuestas de diseño diferentes. En realidad, las metodologías de diseño difieren de una empresa a otra e incluso entre diferentes grupos de la misma compañía. También son muy orientadas a los productos. En este caso, el diseño del sistema por lo general comienza con una plataforma ya definida, por lo general definida por una plataforma bien conocida del proveedor o definidos localmente dentro de la empresa, de igual forma sucede cuando se quiere realizar un nuevo producto, basado en una línea anterior de la empresa. [Gajski, 2009]

En la Figura 14 se observa el esquema de esta metodología donde el inicio parte de la estructura del sistema a la parte física, donde luego por 2 niveles de comportamiento y terminar en los niveles inferiores en los 3 aspectos del sistema.

Esta metodología consta de cinco fases: Exploración, Planificación, Desarrollo, Lanzamiento, y mantenimiento.

Figura 14. Metodología de la plataforma



Gajski, Embedded System Design Modeling, 2009.

En la fase de *Exploración*, los clientes establecen requisitos para el lanzamiento del producto, en esta fase, el equipo de desarrollo identifica la plataforma y las limitaciones de aplicación y las estimaciones del sistema con métricas basadas en las características conocidas del producto. Con esta información a la mano, el equipo de desarrollo estará en condiciones de definir la plataforma del sistema que se utilizará para desarrollar el producto en las siguientes fases.

En la fase de *Planificación*, el propietario de la plataforma y los clientes identificaran más requisitos y priorizaran la cartera de productos, se realizan explicativas del diseño y modificaciones que pueden hacerse en el prototipo, de este modo se clarifican los requisitos del sistema y se organiza la información y procedimientos que más adelante serán desarrollados.

En la fase de *Desarrollo*, los miembros del equipo deben implementar las funcionalidades propuestas en las fases anteriores y programar el sistema basado

en los elementos iníciales, la optimización del sistema también se lleva a cabo durante esta fase. Donde la última versión del sistema programado es la que proporciona el producto que se desplegarán en el entorno operativo.

En la fase de *Lanzamiento*, el producto es instalado y puesto en uso práctico. Durante esta fase, por lo general implica la identificación de errores y la mejora en los servicios del sistema. Por lo tanto, el desarrollador y los clientes deben decidir si estos cambios se incluirán en la versión actual o posterior. Esta fase tiene como objetivo entregar el producto de lanzamiento y la documentación necesaria para el cliente.

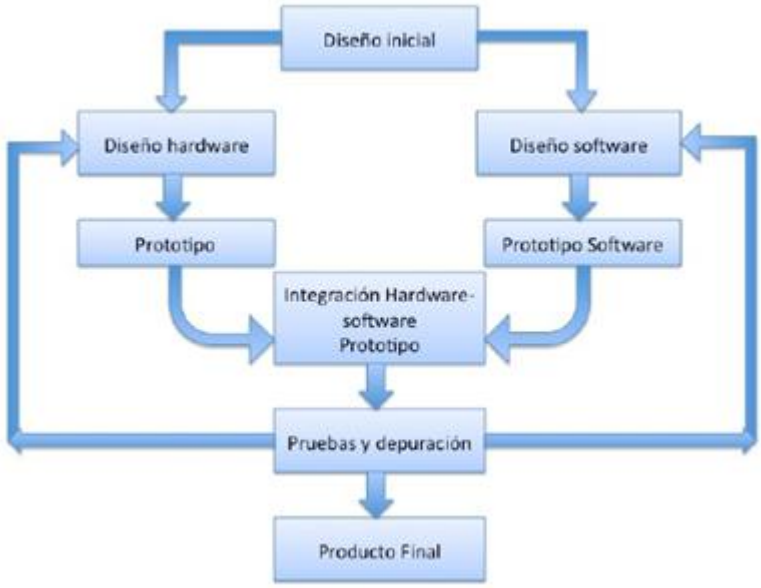
En la fase de *Mantenimiento*, los miembros del equipo deben estudiar el comportamiento del producto a medida que transcurre el tiempo, de este modo pueden observar posibles mejoraras para diseños futuros, así como evaluar la calidad del producto diseñado.
[Cordeiro, 2008]

5.3.5. Metodología de la integración

Esta metodología es utilizada en las empresas donde existen diferentes equipos de trabajo donde las tareas son repartidas, esta metodología pretende realizar individualmente los desarrollos de hardware y software por separado y una vez se tengan todos los componentes listos, se realiza una tarea de integración en la cual se verifica el correcto funcionamiento en conjunto de ambas partes para que luego el producto sea puesto a prueba, de manera general los pasos que sigue esta metodología se muestran en la Figura 15, en donde inicialmente se realiza un diseño inicial del sistema a partir de lo que se va a desarrollar, con base en esto se realiza por un lado el diseño y la fabricación del prototipo de hardware y por otro aparte, el diseño y la programación del software, una vez todo está listo se pasa a una fase de integración donde se unen estos elementos para después realizar pruebas y depuración y así poder obtener un producto final.

Por lo general esta metodología es muy utilizada en el desarrollo de hardware, pues normalmente ese tipo de sistemas son planeados de manera modular, lo cual permite realizar integraciones entre los diferentes componentes, este modelo se apoya en los modelos de ciclo de vida en V y el modelo de ciclo de vida en cascada, lo que permite tener cierta flexibilidad a la hora de hacer modificaciones en el diseño, siempre y cuando se detecten en fases no muy avanzadas a donde ocurrió el error.

Figura 15. Fases típicas de la metodología de integración



Úbeda, Apuntes de Sistemas embebidos, 2009

En la Figura 16 se muestra una aplicación de este modelo de forma muy específica al desarrollo de un sistema embebido utilizando el modelo de ciclo de vida en cascada en donde si bien no se observa el paralelismo entre las etapas de diseño entre hardware y software, la programación no se hace uno sobre el otro, sino que se hace una integración una vez ya están los dos prototipos listos, este modelo es ideal para aplicar esta metodología cuando solo existe un equipo para realizar el diseño del sistema y no es posible realizar tareas paralelas.

Las etapas de esta metodología consisten en:

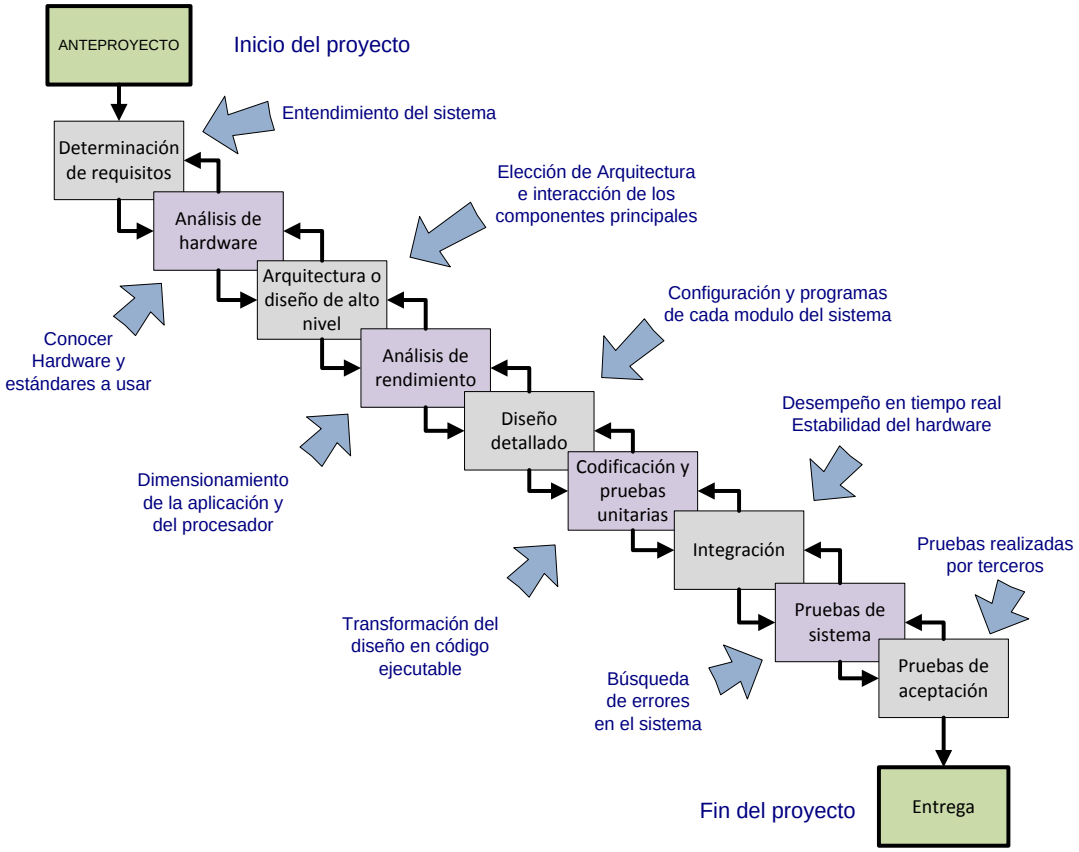
Anteproyecto: En esta etapa se define el proyecto generando la idea de lo que se busca obtener al final, se busca definir la viabilidad, evaluando los diferentes aspectos que componen un proyecto, de este modo se completa el documento de iniciación del proyecto.

Determinación de requisitos: al definir los requisitos de un sistema embebido se obtiene entendimiento común sobre qué es el sistema y qué hace, de igual forma los requisitos sirven de base para evaluar un sistema, los documentos de requisitos son:

- Requisitos de interfaces
- Requisitos de desarrollo
- Arquitectura del sistema
- Matriz de arquitectura vs requisitos
- Contexto y objetivos
- Requisitos funcionales
- Requisitos de prueba
- Matriz de requisitos de prueba vs funcionales
- Requisitos de calidad

Análisis de Hardware: En esta etapa se busca conocer y comprender el hardware que se va a utilizar y los estándares a usar, es muy frecuente utilizar sistemas, herramientas interfaces o estándares, con los que no se está familiarizado, no es posible diseñar sin conocer a fondo las capacidades y limitaciones con que se va a trabajar, puesto que el que pretende aprender durante los “tiempos libres” es ingenuo.

Figura 16. Metodología de la integración



León, La ingeniería de software y los sistemas embebidos, 2010.

Arquitectura o diseño de alto nivel: Crear un modelo del sistema embebido donde se incluyen las componentes principales y sus interacciones, puede ser incluido en el documento de requisitos, la elección de la arquitectura es la decisión de diseño más crítica de todo el sistema, una arquitectura flexible es la base del desarrollo evolutivo, las arquitecturas son difíciles de inventar de la nada, pero se pueden re-usar de proyectos anteriores o de libros.

Análisis de rendimiento: En esta etapa se asegura que el sistema tendrá el rendimiento adecuado, en esta etapa se realizan:

– *Análisis de escenarios*: Determinar operaciones a ejecutar y consumo de recursos por operación

– *Análisis del sistema*: Determinar uso de recursos compartidos entre escenarios.

– *Planificación de tareas*: Determinar qué componentes son tareas y cómo secuenciarlas (*scheduler*).

Diseño detallado: En esta etapa se toman y registran decisiones sobre:

– Responsabilidades de cada módulo

– Interfaces de funciones.

– Algoritmos y/o máquinas de estado

– Estructuras de datos

– Variables y constantes

– Uso de banderas, interrupciones, polling.

El resultado es una descripción detallada de cada módulo del sistema en uno o varios documentos

Codificación y Pruebas Unitarias: En esta etapa se transforma el diseño en código ejecutable, se pretende que la codificación es una actividad poco creativa si el diseño es muy detallado, es más bien una transcripción del diseño al lenguaje de programación, sólo hay que preocuparse del lenguaje y las herramientas, además de encontrar errores de diseño, la codificación de un módulo termina cuando se ejecutan con éxito las llamadas pruebas unitarias.

Integración: Aquí se asegura que no hay errores de interfaces, en los sistemas embebidos la integración suele ser más compleja que en los sistemas comunes, se busca que se cumplan correctamente la concurrencia (semáforos, tareas, dependencias), las operaciones en tiempo real y la estabilidad del Hardware.

Pruebas de sistema: El propósito es asegurar que se cumplen los requisitos, es un proceso muy formal planificado en el Plan de Pruebas de Sistema, se realizan los casos de prueba en la Especificación de Pruebas de Sistema y se consigna los resultados en el Acta de Pruebas de Sistema, cada error encontrado se describe y clasifica en un Informe de Error, la base de datos de errores sirve para tener un registro que sirva de referencia para futuros desarrollos y no cometer siempre los mismos errores.

Pruebas de aceptación: Las pruebas de aceptación buscan certificar que los requisitos del cliente se han cumplido satisfactoriamente y por lo tanto se puede iniciar la producción, esta etapa – también llamada “qualification” – debe hacerse por un equipo independiente del desarrollo, si hay un cliente específico, él puede ejecutar las pruebas, si el producto es para el mercado, debe haber un equipo en la organización o subcontratado para hacerlas.

Entrega: Se entrega al cliente un producto estable y certificado, esto debe ser un acto “formal”, en donde la lista exacta de los entregables con sus versiones debe está identificada en un acta de entrega, los entregables debieron haber sido identificados desde el comienzo del proyecto.
[León, 2010].

5.3.6. Metodología de Codiseño de hardware y software

Para muchos sistemas complejos, el hardware no es más que la plataforma sobre la cual se distribuye el software, con la mayoría de la funcionalidad del sistema implementado en el programa. En tales sistemas, el desarrollo del software es una proporción importante de los esfuerzos de la implementación, puesto que esta es una de las metodologías más usadas y documentadas se hará un poco más de énfasis en sus modelos de desarrollo.

Un aspecto importante de la exploración de la arquitectura es la partición de operaciones entre los componentes de un sistema. El particionamiento es esencialmente la aplicación de la estrategia de resolución de problemas "divide y vencerás". Si los requisitos del sistema implican una serie de pasos de procesamiento, se puede dividir el sistema en un número de componentes, cada uno de los cuales realiza una de las etapas del proceso. Los componentes interactúan entre sí para completar la tarea global del sistema. [Ashenden, 2008]

En los sistemas embebidos hay que tener en cuenta durante su diseño tanto el hardware como el software. Por lo tanto, este tipo de diseño es también llamado codiseño de hardware/software. El objetivo general es encontrar la correcta combinación de hardware y software que resulte en el producto más eficiente de acuerdo a los requisitos. Por lo tanto, los sistemas embebidos no se pueden diseñar por un proceso de síntesis teniendo en cuenta sólo la especificación de comportamiento. Por el contrario, la disponibilidad de los componentes tiene que tenerse en cuenta. También hay otras razones para tener esta restricción: el fin de hacer frente a la creciente complejidad de sistemas integrados y sus estrictos requisitos de tiempo de salida al mercado, la reutilización es esencialmente inevitables.

Debido a la enorme variedad de hardware de los sistemas embebidos, es mucho menos estandarizado que el hardware de los ordenadores personales, la decisión sobre qué partes poner en el hardware y en el software, se denomina *particionamiento*. Hay numerosas ventajas y desventajas a considerar. La funcionalidad implica la evaluación de muchas condiciones y la toma acciones alternativas algunas funciones pueden ser difíciles de implementar en hardware, pero relativamente sencillas en el software. Por el otro lado, las funcionalidades que implican realizar cálculos rápidos en grandes cantidades de datos o los datos que llegan a un ritmo elevado pueden necesitar procesadores de muy alto rendimiento (y por tanto costoso y consumo de potencia), y así puede más

fácilmente ser realizado por el hardware personalizado. Otra consideración es que el software incrustado se puede almacenar en circuitos de memoria que pueden ser reprogramados después de que el sistema se fabrica o despliega. Así, el software puede ser actualizado para corregir errores de diseño o añadir funcionalidad sin necesidad de revisar el diseño del hardware o el reemplazo de sistemas desplegados. [Marwedel, 2006, Ashenden, 2008]

Una manera lineal de seguir esta metodología es la que se muestra en la Figura 17 en donde la realización del sistema se hace a partir de la implementación conjunta de software y hardware, teniendo también en cuenta la validación y evaluación de todo el sistema durante todas las etapas.

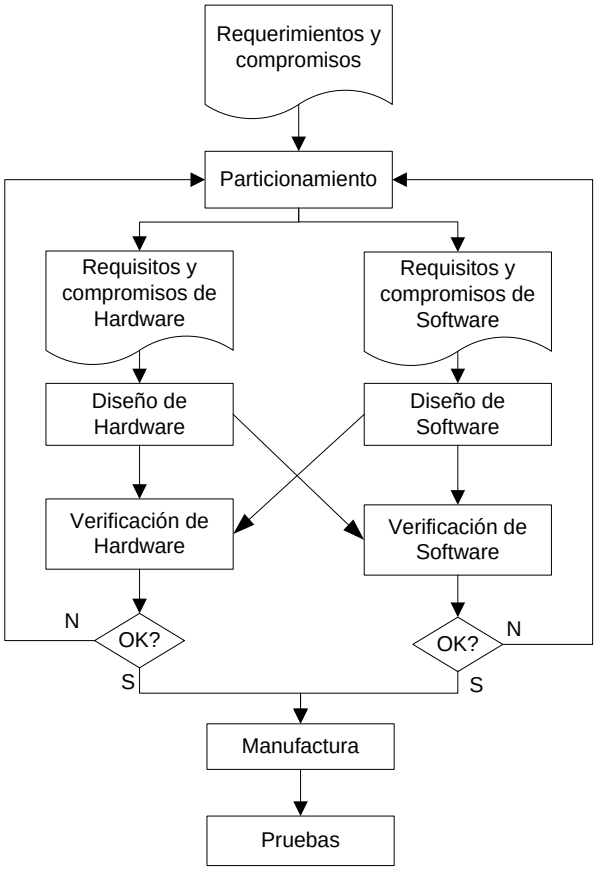
Figura 17. Metodología codiseño lineal

Marwedel, Embedded Systems Design, 2006.

Una vez que la funcionalidad se ha dividido entre el hardware y software, el desarrollo de los dos puede realizarse de forma concurrente, como se muestra en la Figura 18, Para que aquellos aspectos del software embebido que dependen del hardware, los modelos de comportamiento abstractos diseñados a partir del hardware, pueden ser usados para verificar el diseño del software. Esto puede

hacerse utilizando un set de instrucción en un simulador para el núcleo del procesador trabajando conjuntamente con un simulador para el modelo de hardware. Un enfoque similar se puede utilizar para verificar la parte del hardware que funciona conjuntamente con un núcleo de procesador. Los programas de prueba se pueden ejecutar utilizando el simulador de procesador en funcionamiento en conjunto con el simulador de hardware. Los beneficios del desarrollo de hardware y software al mismo tiempo incluyen evitar el tiempo extra que participan en el desarrollo, una tras otra, y la detección temprana de errores en el juego de software y hardware. [Marwedel, 2006]

Figura 18. Una metodología de codiseño



Marwedel, Embedded Systems Design, 2006.

De la Figura 18, se observa que el punto de partida es un conjunto de requisitos y limitaciones. Estos son por lo general generados externamente al equipo de diseño, por ejemplo, el grupo de comercialización de una empresa o un desarrollo de producto para un cliente específico. Por lo general, incluyen requisitos para las funciones (lo que el producto va a hacer), requisitos de desempeño (qué tan rápido lo hace), y las restricciones sobre el consumo de energía, costo y empaque.

La metodología de diseño especifica tres tareas:

- Diseño funcional
- Síntesis
- Aplicación física.

Cada uno de ellos es seguido por una función de control. (Las tareas de diseño y de verificación funcional se exponen para indicar que son en realidad un poco más complicados de lo que se muestra en el diagrama) Si la verificación falla en cualquier etapa, es necesario revisar una tarea previa para corregir el error. Idealmente, solo se debería volver a visitar sólo la tarea inmediatamente anterior y hacer una pequeña corrección. Sin embargo, si el error es lo suficientemente grave, es posible que sea necesario retroceder aún más para hacer cambios más significativos. Por lo tanto, al realizar una tarea de diseño dado, vale la pena tener en cuenta las limitaciones que existían en las tareas siguientes, de forma que se evite la introducción de errores que se detectan más tardíamente. Una vez que las tareas y las actividades de verificación se han realizado, el producto puede ser fabricado, y cada unidad de prueba para asegurarse de que es funcional, la Figura 19 presenta la metodología para el codiseño de un sistema compuesto por componentes de hardware y software. [Nunes, 1998]

Figura 19. Etapas del codiseño

Nunes, Hardware-Software Codesign of Embedded Systems, 1998.

Hay cuatro tareas a realizar en el diseño integrado de hardware y sistemas informáticos:

- **Especificación y análisis:** El sistema se describe utilizando lenguajes visuales y/o de texto, en diferentes niveles de abstracción. Esta descripción es analizada a fin de conocer si los componentes deben ser implementados en hardware o software.

- **Separación**: El sistema se divide en sus dos componentes elementales, hardware y software. Siempre que hay un flujo de datos entre los diversos componentes de hardware y software se debe proporcionar una interfaz.

- **Síntesis**: Componentes de hardware y software aplicadas en sus primitivas básicas (las compuertas lógicas de hardware y el lenguaje de máquina para el software) y una interfaz que se genera para las diferentes interconexiones entre ellos.

- **Validación**: Antes de producir un prototipo, un modelo virtual del sistema es construido y probado en un equipo. Existen diferentes técnicas disponibles para validar un diseño que van desde verificaciones formales, la simulación y la emulación.

5.3.7. Metodologías mixtas o propias

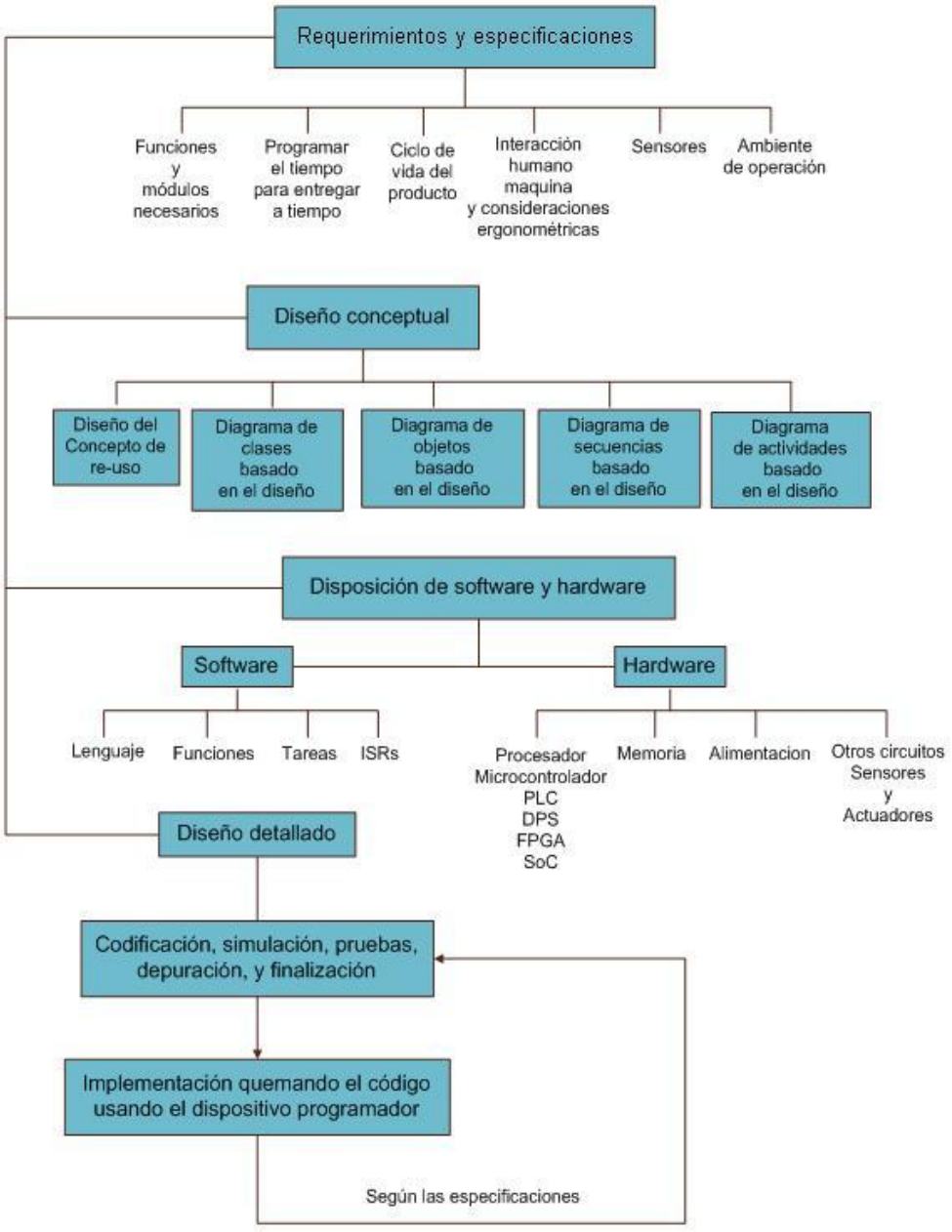
Como se mencionó anteriormente, las empresas de productos electrónico de consumo tienen sus propias metodologías de sistemas desarrolladas a partir de las funciones de sus productos, una empresa de sistemas de medición debe enfocar sus esfuerzos en la parte de precisión de la medición, mientras que una empresa de celulares se preocupa más por un buen desarrollo de las interfaces de los usuarios, por este motivo, una metodología estándar no está en capacidad de cubrir todas la necesidades de los mercados, gracias a este problema, se encuentran una gran variedad de metodologías de diseño en la industria con diferentes enfoques, ya sea al hardware, al desarrollo de software e interfaces o al concepto del producto como tal, y dejando en un tercer plano las partes del sistema que no están en el núcleo de la funcionalidad del producto.

A continuación se muestra como ejemplo una metodología de diseño propuesta por el autor Raj Kamal [2008], en donde es tomada una parte de la metodología de integración para diseñar el concepto del producto y hacer el derrotero de actividades a seguir, y para la parte del desarrollo y configuración del producto hace uso de la metodología de codiseño, el diagrama de etapas de esta metodología se muestra en la Figura 20 principalmente esta metodología plantea cuatro etapas que son:

- Toma de requerimientos y especificaciones del sistema
- Diseño conceptual
- Disposición de hardware y software

- Diseño detallado.

Figura 20. Plan de acción para el diseño de un sistema en su proceso de desarrollo



Kamal, Embedded systems: architecture, programming and design. 2008.

Requerimientos del sistema y especificaciones: El primer paso es tener una completa claridad acerca de las especificaciones del sistema que se necesita.

- Funciones y tareas del producto: Entender las funciones y tareas que el usuario necesita es esencial, considere el diseño de un sistema embebido para un robot, ¿cuáles son las funciones que se esperan de él?, ¿cuáles son los grados de libertad requeridos para el movimiento de sus partes?, ¿cuáles son las tareas que debe cumplir?, etc.

- Tiempo de entrega programado: El tiempo de entrega programado es importante, tiempos de entrega muy apretados forzara a utilizar modelos de desarrollo de altas velocidades dejando a un lado el modelo de desarrollo lineal idóneo o forzando a usar una combinación orientada a objetos utilizando herramientas de cuarta generación utilizando procesadores de propósito general.

- Ciclo de vida del producto: Cuando el ciclo de vida de un producto es muy corto, se necesitaran frecuentes cambios en el diseño del sistema por parte del equipo de diseñadores.

- Carga en el sistema: La carga del sistema es una especificación muy importante, el sistema puede fallar por sobrecarga, por ejemplo en el hardware y en el procesador de un procesamiento de imágenes, un video clip o un video en tiempo real tendrá diferentes cargas que podrían sobrecargar el sistema si no está bien dimensionado.

- Interacciones humano máquina: Las especificaciones son necesitadas para responder esas preguntas, cuál será la salida en la pantalla del sistema, esto es la conducta para las entradas por teclado y las salidas de

visualización, por ejemplo , esto se refiere al teclado de un controlador remoto de un PC.

- Entorno de operación: Temperaturas de operación, humedad y especificación de los parámetros son esenciales, un sistema puede fallar en las montañas o puede fallar operando a altas temperaturas.

- Sensores: Especificaciones de los sensores para la sensibilidad, resolución y precisión, son esenciales para el diseño según los requerimientos.

- Requerimiento de energía y medio ambiente: Un sistema que tenga una gran carga, necesitara grandes potencias, un sistema dependiente de baterías necesitara soluciones de manejo de potencia tales como diseño de hardware y software con ahorro inteligente de energía, un sistema operando bajo condiciones de disponibilidad continua de potencia tendrá especificaciones diferentes, el diseño es más simple en los casos donde no se necesita memoria para guardar el estado del sistema.

- Costo del sistema: Los costos máximos manejables deben ser especificados para decidir si un proyecto es aceptable para un equipo de desarrollar y la cantidad de esfuerzo que debe hacer el equipo.

Diseño conceptual: El segundo paso es desarrollar un diseño conceptual del sistema, el diseño conceptual ayuda en el desarrollo de estructura y diseño de aplicaciones de software y hardware, este diseño del modelo conceptual puede ser desarrollado usando un enfoque UML (*Unified Modeling Language: Lenguaje Unificado de Modelado*), es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad, Es importante resaltar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para

documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo, los diagramas UML han ido evolucionando a través del tiempo y hoy en día se encuentra en la versión 2.0, está compuesto por una serie diagramas las cuales buscan definir completamente todo el modelo, entre los diagramas más comunes utilizados en UML se encuentran:

- Diagrama de Casos de Uso
- Diagrama de Clases
- Diagrama de Objetos
- Diagrama de Secuencia
- Diagrama de Colaboración
- Diagrama de Estados
- Diagrama de Actividades
- Diagrama de Componentes
- Diagrama de implementación

Diseño de la disposición de software y hardware: El tercer paso es el desarrollo de la estructura del hardware y software del sistema, existen dos enfoques:

- Diseño independiente enfocado a ser seguido por la integración: Una vez se termina el diseño del software, se inicia con la integración al hardware aprovechando que todo el sistema está diseñado desde la etapa anterior.
- Enfoque de codiseño concurrente: este enfoque es necesario para sistemas embebidos sofisticados, en donde ambos ciclos de vida (Hardware y Software) proceden concurrentemente lo cual disminuye los tiempos críticos.

Especificación de herramientas para la implementación de hardware y software.

La cuestión principal antes de desarrollar un esquema y diseño detallado es el siguiente: ¿cuáles son los elementos (Hardware y software requeridos) para el proceso de desarrollo?, la respuesta puede ser dada seleccionando los elementos necesarios para cumplir las especificaciones obtenidas.

Diseño detallado: El cuarto paso es el diseño detallado de los códigos y tarjetas del sistema seleccionando primero el procesador y la memoria, entonces se decide acerca de las funciones que van a ser implementadas en el hardware y el software. El esquema de software y hardware ayudara luego en el diseño detallado para la implementación de códigos de software detallados y el circuito para obtener la tarjeta del sistema.

Una vez se ha terminado el diseño detallado del software y del hardware, se procede a la implementación de códigos en el lenguaje y herramienta de programación elegida, de igual manera se fabrica el prototipo de hardware del sistema, una vez se tengan estos elementos se inicia un ciclo en el cual se realizan pruebas del funcionamiento del software y se someten a una serie de revisiones de modo que todo cumpla con las especificaciones iníciales del sistema, para obtener así el producto final.
[KAMAL, 2008].

5.3.8. Tabla de metodologías de diseño de sistemas embebidos

En la Tabla 4 se realiza una recopilación de las ventajas y desventajas de las metodologías anteriormente descritas, con el fin de tener un punto de comparación para la selección de alguna aplicación, según el tipo de producto a desarrollar.

Tabla 4. Comparación de las metodologías de diseño de sistemas embebidos

Metodología	Ventajas	Desventajas	Aplicaciones más comunes
Bottom-Up	Se abordan inicialmente los problemas físicos, lo cual produce buena robustez a nivel de hardware	La corrección de errores en una etapa avanzada es muy difícil y costosa.	Pequeños sistemas modulares de control propensos a ampliaciones, sistemas poco complejos.
Top-Down	Se delimitan los alcances del proyecto para luego ir a lo específico teniendo así una meta fija.	El cambio en los requerimientos una vez iniciado el diseño hace que sea muy costoso realizar cambios en el sistema.	Aplicaciones de control específicas, sistemas de complejidad media-baja que no llevan mucho tiempo de desarrollo.
Meet in the middle	Aplica las dos metodologías anteriores en los procesos de diseño según sea conveniente manejarlos de lo más grande a lo más pequeño o viceversa	El manejar las dos metodologías anteriores puede tener un tiempo de desarrollo mayor si no se define bien que cosas se van a tratar con cada metodología.	Aplicaciones de complejidad media que no tengan un time-to-market muy pequeño
Basada en la plataforma	Se aprovechan las plataformas de desarrollo y hardware de la empresa	Se limita a las prestaciones que posea el hardware	Productos con el mismo hardware pero con diferentes modelos y mejoras, donde la plataforma es la misma pero las aplicaciones cambian.
Integración	Varios equipos de trabajo pueden ir desarrollando paralelamente el sistema.	Si la etapa de diseño no es muy rigurosa se pueden presentar grandes problemas a la hora de integrar el SW en el HW	Es utilizado en las multinacionales como Sony o LG para el diseño de algunos electrodomésticos.

Tabla 4. Comparación de las metodologías de diseño de sistemas embebidos (continuación)

Co-diseño	**Da la posibilidad de detectar errores en etapas tempranas de diseño tales como incompatibilidades de hardware o problemas de rendimiento.**	**Los resultados del proyecto solo se observan al final del proceso.**	**Normalmente es el método más utilizado académicamente en el desarrollo de sistemas embebidos, muy útil en la construcción de sistemas de mediana y alta complejidad.**
Mixtas o Propias	Permite ajustar las etapas a desarrollar teniendo en cuenta el tipo de producto, que no necesariamente están plenamente cubiertas en los métodos comunes.	La elección de etapas no apropiadas para el tipo de aplicación puede incurrir en demoras innecesarias en el desarrollo del sistema	A nivel empresarial utilizan sus propios métodos obtenidos a partir de la experiencia con el desarrollo de sus propios productos.

Elaboración propia.

Basado en toda la revisión bibliográfica acá presentada se establece un protocolo de diseño a partir del cual sea posible desarrollar sistemas embebidos destinados a la medición de variables, buscando la detección temprana de los errores en el diseño y la cual sirva como guía para la creación de nuevos productos electrónicos que brinden la mayor satisfacción al usuario final.

En este método se busca seguir las 4 etapas principales que dicta el modelo de ciclo de vida de un producto los cuales son:

- Toma de requerimientos y planeación
- Análisis y diseño
- Desarrollo e implementación

- Pruebas y evaluación

Inicialmente se definen una serie de generalidades del tipo de sistema a desarrollar, con base a estas será posible la elección del modelo de ciclo de vida a seguir, ya sea el modelo en cascada, modelo V o modelo evolutivo, dicho modelo de ciclo de vida será el patrón a seguir por el protocolo a desarrollar de modo que utilizando etapas de las diferentes metodologías presentadas, sea posible desarrollar un prototipo de un sistema embebido en el menor tiempo posible y que permita cierta flexibilidad en el manejo de los errores que se puedan encontrar en las diferentes etapas del proceso, teniendo en cuenta además los tiempos de demora en la gestión de componentes y demás procesos asociados a la tramitología de los proyectos.

5.4. GENERALIDADES DEL SISTEMA EMBEBIDO

La definición inicial del producto a desarrollar se obtiene en la primera fase del método a través de la toma de requerimientos al usuario, sin embargo es necesario definir a que conjunto específico de los sistemas embebidos debe apuntar este método de desarrollo, como se menciona en capítulos anteriores, el diseño de un sistema de control automático en un proceso industrial es muy diferente a la de un dispositivo cuyo fin sea solo interactividad del usuario.

Producto a desarrollar:

De manera técnica lo que se busca es realizar un sistema embebido que permita la adquisición de datos de un sensor que sirve de transductor con el medio ambiente, de este modo se debe buscar que el sistema posea canales de entradas análogos digitales y se vuelve prioritario darle el mejor manejo posible a esta etapa para que el sistemas posea una buena precisión y confiablidad con respecto a los datos que vaya a procesar.

Por otro lado el sistema deberá contar con una interfaz HMI (Human-Machine interface) la cual debe ser intuitiva en el manejo para el usuario, de este modo el sistema a diseñar también deberá tener puertos de entrada para que el usuario pueda ingresar y elegir información en el sistema, así como también una interfaz de salida para un módulo de visualización y un módulo de almacenamiento por medio del cual, el usuario pueda llevar los datos capturados a una computadora, también es de tener en cuenta que la programación de interfaces en módulos de visualización requieren de buena cantidad de memoria de almacenamiento y buen tiempo de desarrollo de las tramas y objetos a dar visualización en pantalla.

El sistema debe ser portable, por lo tanto es necesario tener en cuenta desde un principio el manejo del consumo de potencia de modo que la programación y conexión de todos los elementos del sistema estén orientadas al bajo consumo.

El sistema a desarrollar es un sistema que busca cubrir una necesidad específica y puntual, de este modo para un primer prototipo no deberían de existir cambios en los requerimientos iníciales hasta que se realicen las primeras pruebas lo cual debe permitir trabajar un modelo específico definido inicialmente.

5.5. ELECCIÓN DE UN MODELO DE CICLO DE VIDA

Con base a los anteriores aspectos se obtienen algunas características semejantes a las que posee el ciclo de vida en cascada, debido a que el prototipo se define totalmente en el inicio del desarrollo y no esta propenso a cambios como sucede en algunos desarrollos de la industria que son manejados por los estudios de mercadeo, ni tampoco es un proceso evolutivo de etapas donde se necesita un prototipo funcional en cada una.

De igual manera el hecho de que la parte de adquisición de datos sea fundamental para que el equipo sea confiable en sus Mediciones se necesita que esta etapa

funcione perfectamente antes de proseguir con las otras. Paso que es favorecido en el modelo de ciclo de vida en cascada donde las etapas se van completando una a una y solucionando los problemas que en estas se presenten, si se eligiera un modelo de desarrollo basado en la integración podría encontrarse el problema de que el sistema trabaje correctamente con señales simuladas, pero que a la hora de ser integrado el sistema, no trabaje adecuadamente con las señales físicas ya sea por problemas de diseño o problemas electromagnéticos tales como el manejo de tierra, etc. por lo tanto, el modelo elegido para el protocolo es el modelo de ciclo de vida en cascada.

6. PROTOCOLO DE DESARROLLO

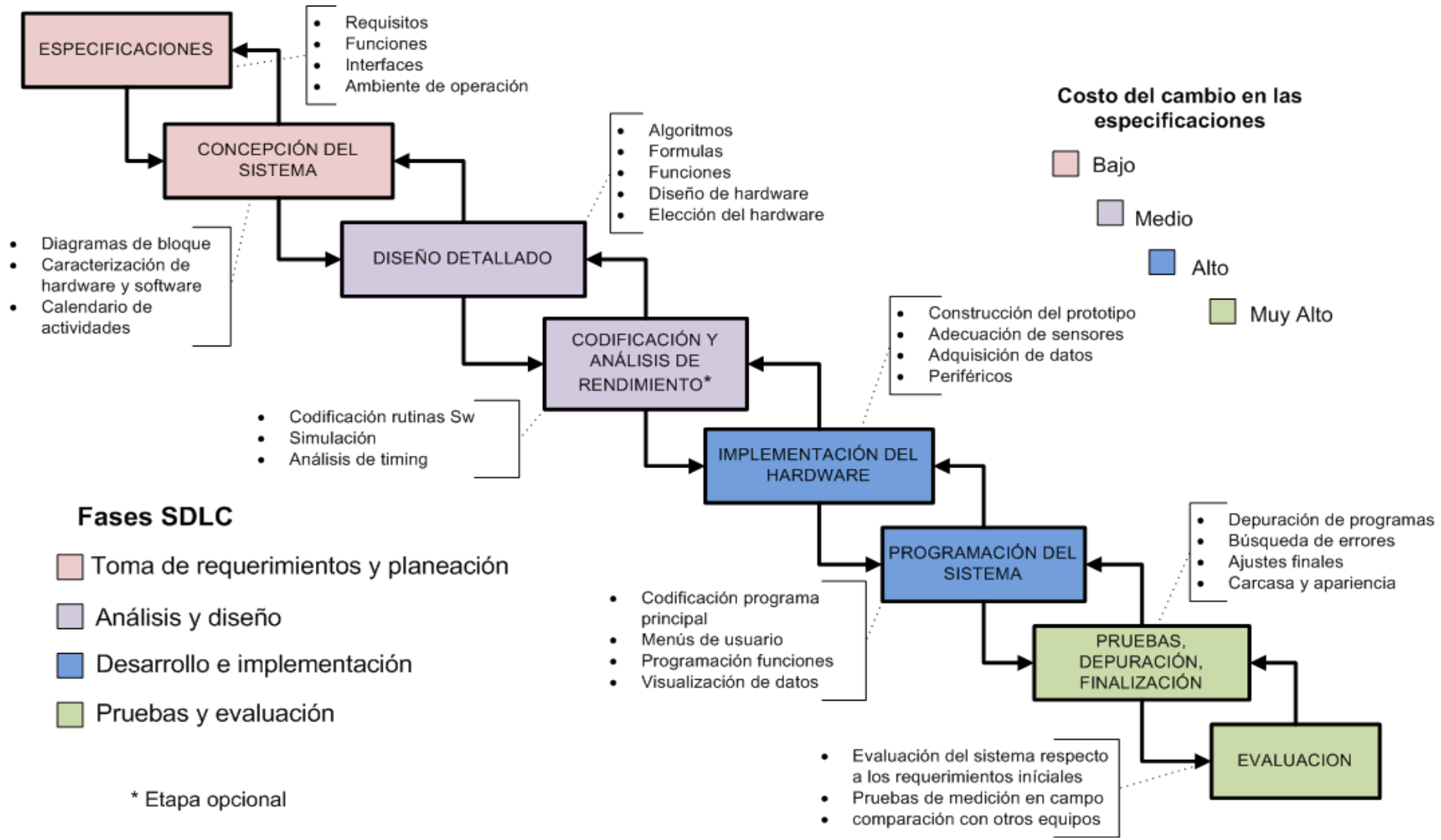
A continuación se describen los pasos del protocolo a seguir para el diseño y construcción del sistema embebido, se propone seguir un modelo de desarrollo en cascada que contiene etapas de la metodología de *integración*, etapas de la metodología del *codiseño* y etapas donde se aplica el diseño *Top-Down*, de modo que en las primeras etapas se haga un diseño completo del sistema en papel, partiendo desde la funcionalidad general del sistema, teniendo en cuenta todo el posible procesamiento que deba tener el sistema, de igual forma se busca analizar en simulación el timing del sistema para conocer si el diseño de hardware propuesto en el inicio si cumple con las expectativas esperadas, o si por el contrario se hace necesario acudir a procesadores más costos y de mayores capacidades (*Integración*). Luego de tener muy claro todos los algoritmos y procesos involucrados en el sistema, se busca seguir con las etapas de *codiseño*, donde la correcta adquisición de datos provenientes del ambiente es vital para el sistema, por lo cual en esta parte se busca hacer un codiseño entre la programación que lee las señales y el hardware de acondicionamiento involucrado en esta etapa, una vez la etapa de adquisición esta lista, se aplican los algoritmos de procesamiento y se realiza la parte de visualización y navegación del usuario con la cual va a tener acceso a los datos, por ultimo ya cuando el sistema esté completo, se entra en una etapa de realización de pruebas y evaluación, donde se hace una búsqueda de errores y depuración, para que posteriormente el prototipo sea evaluado por un tercero el cual dará su punto de vista si el sistema cumple o no con los objetivos enunciados al principio del desarrollo.

Se recomienda generar un documento completo para cada etapa del protocolo, en el que se consignan todos los factores y detalles analizados en cada fase, esto permite llevar un registro completo del sistema, ayudando en la documentación final, permite una evaluación más clara de la calidad del sistema, así como ser la

base para futuras mejoras del sistema desarrollado y ayudar en los procesos de reciclaje de software para otros proyectos.

En la Figura 21 se observa el esquema del protocolo propuesto para el diseño y construcción del prototipo, en otros documentos adjuntos a este se encuentran las descripciones completas de las actividades a realizar en cada una de las etapas del sistema.

Figura 21. Protocolo de desarrollo de un sistema embebido



6.1. ESPECIFICACIONES

La etapa inicial de protocolo propuesto busca capturar todas las ideas que están en la mente del usuario, con el fin de poder brindarle la mayor satisfacción con el producto a desarrollar, es muy común que cuando los diseñadores comienzan a diseñar e implementar soluciones que no han sido completamente especificadas conduce a la construcción de sistemas que no satisfacen las necesidades de los clientes y que incurren en el aumento de los costos y en el incumplimiento de los plazos establecidos, en el campo de los sistemas embebidos dicha problemática general de los sistemas de información no solo se conserva sino que se agrava hasta el punto que el 60% de las componentes que integran hardware y software deben ser rediseñadas luego de haber sido programadas. La problemática expuesta para los sistemas en general, tiene mayor impacto en el caso de los sistemas embebidos.

Esta etapa consiste en una entrevista con el usuario en donde él expresa sus ideas respecto al sistema a desarrollar, posteriormente se le realiza una serie de preguntas sobre temas específicos para aclarar algunos puntos de interés para el desarrollador a partir del cual se genera un documento de requisitos que será la guía para el desarrollo del sistema y la herramienta de evaluación del proyecto final, se busca entonces recoger todas las características que debe tener el sistema embebido final, puesto que este protocolo esta propuesto para que todos los requisitos del sistema queden bien definidos al inicio del desarrollo y no se realicen cambios durante la ejecución, se vuelve de vital importancia la recolección de todos los requisitos.

Como primer paso del protocolo, en la entrevista se deben consignar todas las ideas que posee el usuario acerca del sistema a desarrollar, de modo que él pueda expresar en sus propias palabras cuáles son sus necesidades y cuáles son las características que el sistema debe tener buscando que todo sea lo más claro y detallado posible.

El cuestionario que se realiza al usuario, toma las características más relevantes de la literatura de ingeniería de requisitos para sistemas embebidos, tomando en cuenta que hay que utilizar las que se refieren a la recolección de datos del ambiente a través de sensores y los que se refieren a interfaces humanas pues de esta última depende mucho la total satisfacción del usuario, este cuestionario se encuentra en el ANEXO B. Entrevista de captura de requisitos.

Dicho cuestionario trata sobre temas puntuales con los que se busca profundizar y aclarar diversos temas sobre la finalidad, el funcionamiento y desempeño del sistema embebido que se quiere construir, entre estos temas están:

- Disponibilidad
- Acceso
- Entrada-respuesta
- Interacción de agentes
- Funcionalidades
- Desempeño
- Precio y costos
- Consumo de energía

Con estas preguntas se busca que el usuario pueda expresar en palabras ordinarias el comportamiento que desea de todo el sistema, de este modo se busca que a partir de estas características mostradas por el mismo usuario, sea posible realizar un producto que lo deje satisfecho.

Una vez es finaliza todo este proceso con el usuario, se procede a realizar una recopilación de la información adquirida haciendo una lista numerada de los aspectos del sistema concernientes a los elementos de la siguiente tabla:

Lista de requerimientos del sistema:

- Funcionalidad
- Composición de hardware
- Parámetros de operación
- Parámetros de sensores
- Características físicas
- Aspectos misceláneos relevantes

Funcionalidad

Esta lista debe contener las funciones o tareas que el sistema a desarrollar debe estar en capacidad de realizar.

Composición de hardware

La lista referente a este ítem debe contener todos los aspectos relacionados con la composición del hardware del sistema, tales como los módulos que debe tener el sistema, diversos tipos de componentes, si está compuesto por una seria subsistemas o en si hace parte de otro sistema mayor, etc.

Parámetros de operación

En este ítem se deben consignar todos los aspectos relacionados con las métricas del sistema, esto se refiere a la hoja de datos característica que tendría el prototipo final, tales como velocidad de procesamiento, tiempo de respuesta, sensibilidad, precisión, etc.

Parámetros de sensores

En caso de que el sistema posea algún sensor o transductor de entrada, cuáles serían las características específicas que estos elementos deben tener según la aplicación.

Características físicas

Según el usuario cuáles son los aspectos fiscos del producto a desarrollar, tales como tamaño, peso, carcasa, protección respecto al ambiente de operación, etc.

Aspectos misceláneos Relevantes

Lista de otros aspectos que son importantes para el desarrollo y no han sido contemplados en los ítems anteriores, tales como costos, tiempo de desarrollo, características especiales, etc.

Una vez generada esta lista, se debe consignar en un documento que reúna además los acuerdos y compromisos para aspectos tales como:

- Aspectos relacionados con el ciclo de vida que no están incluidos en este protocolo tales como mantenimiento, uso y desecho del sistema a desarrollar
- Compromisos posteriores a la entrega del prototipo final
- Documentación requerida del sistema
- Aspectos legales del sistema (protección de propiedad intelectual)
- Sobrecostos en el cambio de las especificaciones
- Planos eléctricos y de hardware en general
- Programas del sistema
- Lista de entregables al final del proyecto

Una vez se ha llegado a común acuerdo sobre este documento de especificaciones, se debe firmar un acta donde se manifiesta la aprobación del desarrollo del sistema basado en las características consignadas en dicho documento.

Este documento será la guía fundamental para generar el inicio y el desarrollo de todo el sistema, asimismo servirá como herramienta de evaluación del sistema

final, de modo que se pueda verificar que el producto desarrollado si cumple con todas las especificaciones solicitadas por el usuario al inicio del proyecto.

Actividades de la etapa:

- Captura de la idea
    - Descripción de la idea y las necesidades del usuario
    - Encuesta de aspectos específicos
- Redacción de documento de especificaciones
    - Lista de requerimientos
    - Lista de compromisos y acuerdos
- Firma del acta de aprobación e inicio.

6.2. CONCEPCIÓN DEL SISTEMA

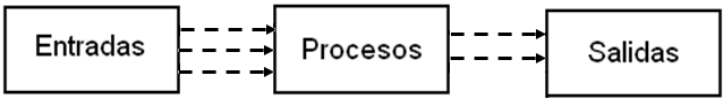
A partir del documento de especificaciones del sistema obtenido en la etapa anterior, se busca hacer un diseño conceptual partiendo de la metodología Top-Down en donde se hace un bosquejo o diagrama de bloques generales de la composición del hardware y software del sistema.

Se hace necesario conocer primero las necesidades y el objeto de aplicación del sistema que tiene el usuario, en caso de que el usuario manifieste necesidades de comparación, de utilización de normas, o algunas otras actividades de este tipo, se debe reunir la información necesario, para poder estar en capacidad de hacer elecciones y diseños basados en el conocimiento del campo de aplicación requerido, una vez realizadas dichas tareas, se procede entonces a la concepción como tal del sistema.

Diagrama de caja negra

Se hace útil iniciar el análisis del sistema a través de la vista de caja negra, en donde se destacan cuáles son las entradas que tiene el sistema y cuáles son los datos o señales que debe entregar el usuario y a partir de eso, generar los flujos requeridos para cumplir las funciones necesarias, como se muestra en la Figura 22 este diagrama suele ser muy sencillo:

Figura 22. Diagrama de caja negra del sistema

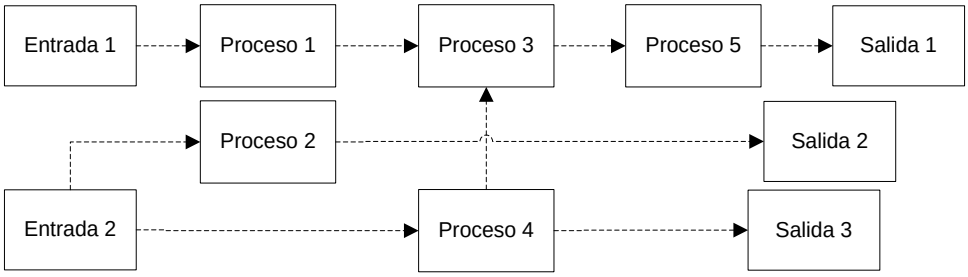


Elaboración propia

Diagrama de flujo de información del sistema

Conociendo las entradas y salidas del sistema se busca hacer un diagrama del flujo de la información, de modo que en forma de diagrama de bloques se muestre las diversas transformaciones o procesos que sufren los datos desde que ingresan al sistema hasta que se generan las salidas que necesita el usuario, se deben tener en cuenta todos los aspectos del documento de especificaciones, tales como composición y funcionalidades del sistema, además de las respuestas correspondientes al tema de *Acciones del agente*, entregadas por el usuario en la entrevista de captura de requisitos de la primera etapa, la Figura 23 muestra un ejemplo de cómo debe ser este diagrama.

Figura 23. Diagrama de bloques del flujo de la información del sistema



Elaboración propia

Diagrama de software del sistema

Una vez se conocen todos los procedimientos que se necesitan en el sistema, se prosigue con la división de los procesos, definiendo cuáles deben o pueden ser cumplidos con hardware y cuáles deben ser generados a partir de programas de software, una vez separados estos aspectos, se pasa a hacer el diseño conceptual del software, pues a partir de este, es posible conocer la capacidad de procesamiento que debe tener el hardware, para este diseño de software se busca realizar un diagrama de flujo (ver ejemplo Figura 24) en el cual pueda apreciarse la ejecución que deberá tener el sistema, si es una máquina de estados, que tipo de procesos se ejecutan en cada estado y cuáles serían las transiciones entre estado y estado, el detalle de este diagrama va hasta mostrar que función se debe ejecutar en cada etapa por ejemplo: aplicación de filtro digital, visualización en pantalla, lectura de sensor, etc. de igual forma a partir de este diagrama se obtiene una serie de funciones necesarias que deben ser desarrolladas, estos diagramas se hacen a partir de la sección *disponibilidad de objetos* de la encuesta de requisitos y con base en los ítems de funciones del sistema y composición de hardware de la lista de especificaciones realizada en la etapa de especificaciones.

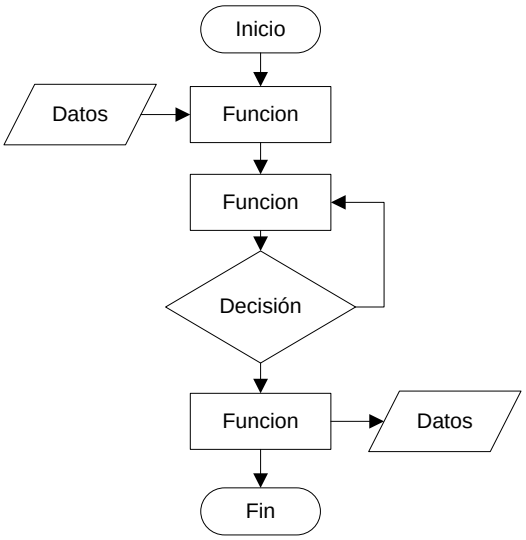
En esta etapa del diseño de los diagramas de bloques del software del sistema, se debe tener en cuenta la calidad del software, en donde se debe buscar que el flujo del programa sea modular y el programa principal sea a partir de llamado de

rutinas o funciones, de este modo se hace más sencillo seguir el comportamiento del programa, detectar errores, reciclar código y hacer posibles actualizaciones al programa desarrollado.

Diagrama de hardware del sistema

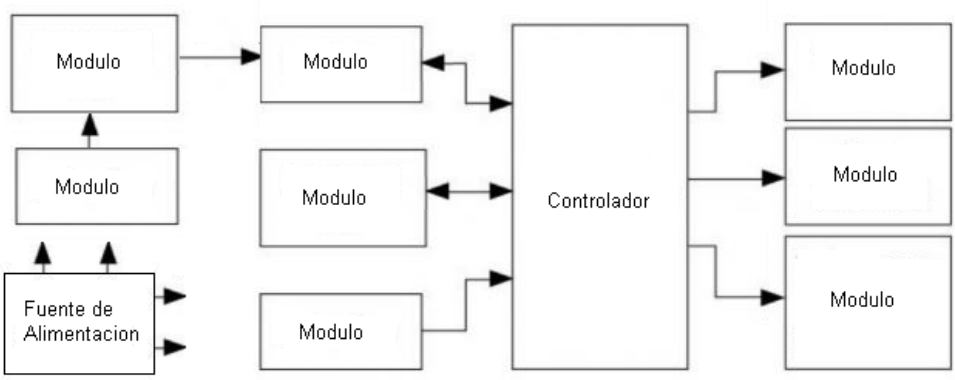
En el diagrama de bloques de hardware se busca representar el flujo de la información, tales como acondicionamiento de señales, adquisición de datos, procesamiento, almacenamiento, visualización, etc. de modo que se pueda observar los módulos de hardware involucrados desde que ingresa la información al sistema, hasta que es procesada y entregada (ver ejemplo Figura 25). Las etapas del hardware se crean a partir de los ítems de funciones del sistema y composición de hardware de la lista de especificaciones realizada en la etapa de especificaciones.

Figura 24. Ejemplo del diagrama de flujo del programa



Elaboración propia

Figura 25. Ejemplo diagrama de bloques de hardware



Elaboración propia

Más adelante en la etapa de Diseño detallado se elegirá el hardware específico que se utilizará, por ahora todo el hardware del sistema estará representado en dicho diagrama de bloques del cual se obtiene una lista de módulos a diseñar y elegir en la siguiente etapa.

Carcasa del sistema

Otro aspecto que se debe iniciar en esta etapa, es la de la visualización del sistema, si el usuario consigno la idea de la forma física o si el sistema debe tener una carcasa que lo proteja del medio ambiente, se deben empezar a hacer los bosquejos de cómo debería ser su distribución o forma externa, de modo que a cualquier duda se pueda ir consultando al usuario y además es un elemento útil a tener en cuenta en la siguiente etapa para la distribución de los componentes en función a su disposición final.

Si en algún punto de este diseño se encuentra algún vacío o inquietud sobre el funcionamiento del sistema, es posible aun retroceder a la etapa anterior y hacer una retroalimentación con el usuario sobre el funcionamiento del sistema, de este

modo es posible realizar el diseño más completo y buscando siempre la satisfacción del interesado.

Cronograma de actividades del proyecto

Una vez se construyen estos diagramas y listas, se puede tener una concepción global del sistema, así como una valoración estimada del tiempo y la complejidad que puede llevar cada tarea del sistema. A partir de este enfoque será posible realizar un cronograma de actividades que sirva como guía para el resto de proceso de diseño, puede ser útil realizar este cronograma a partir de las valoraciones de tiempo que se le puede dar al desarrollo de cada bloque, y al tiempo que tomaría realizar la integración de estos al sistema, así mismo sería posible conocer que actividades del cronograma se podrían delegar, de modo que se pueda avanzar más rápidamente en el proyecto, esto es posible, dado que con un buen diseño es posible asignar tareas, por ejemplo la codificación de un algoritmo puede ser realizada por cualquiera, siempre y cuando se haya hecho un diseño bien completo y detallado de su funcionamiento. Debido a esto hay que tener en cuenta que los diseños se deben realizar muy a conciencia buscando que sea posible delegar tareas.

Otro aspecto a tener en cuenta durante la generación del cronograma de actividades, es la asignación de un tiempo considerable a la etapa de pruebas depuración y finalización, puesto que en esta etapa se debe probar y evaluar el sistema en todos los aspectos específicos, dichas pruebas llevan una gran cantidad de tiempo, y puede ocurrir que por tener tiempos ajustados en esta etapa y no se hagan muchas pruebas, el evaluador final encuentre aspectos no deseados, que si bien pueden ser muy fácil de corregir, no dejan una buena imagen del sistema.

Todos los aspectos de diseño y cronogramas generados en esta etapa, deben ser consignados en un documento detallado que va ser la guía para la próxima etapa

y que será también de mucha ayuda en la documentación final del proyecto y como referencia para futuros desarrollos.

Actividades de la etapa:

- Realizar diagrama de caja negra del sistema
- Realizar diagrama de flujo de la información en el sistema
- Realizar diagrama de software del sistema
- Realizar diagrama de hardware del sistema
- Iniciar con la parte visual de la carcasa del sistema
- Realizar cronograma de actividades del proyecto teniendo en cuenta los tiempos de desarrollo y ejecución de cada etapa

6.3. DISEÑO DETALLADO

En esta etapa se busca diseñar en detalle cada elemento del sistema expresado en los diagramas de bloques de la etapa anterior, una de las características principales de los sistemas embebidos, es que son de más bajo costo que las computadoras tradicionales porque solo utilizan la capacidad de procesamiento necesaria para la tarea específica para la que es diseñado, siguiendo este precepto, se hace necesario realizar primero el diseño en detalle de toda la estructura del software, pues en base al nivel de procesamiento que este necesita, será la capacidad del procesador y memoria del sistema elegido.

Desarrollo de los módulos de software

El diseño detallado del software, va hasta el diseño de las rutinas en pseudocódigo de cada función o bloque mostrado en la etapa anterior y no hasta la codificación en el lenguaje como tal, puesto que el lenguaje y las rutinas van a depender de la arquitectura de la plataforma elegida.

Figura 26. Ejemplo algoritmo en pseudocódigo

```
En-caso-de <expresión> haga
  Caso <opción 1>:
        <instrucciones>
  caso <opción 2>:
        <instrucciones>
  caso <opción 3>:
        <instrucciones>
...
  caso <opción N>:
       <instrucciones>
SINO <instrucciones a realizar si no
        se ha cumplido Ninguna de
        las condiciones anteriores>
Fin-Caso
```

Elaboración propia

Según el tipo de aplicación, se debe buscar que la ejecución del código del software sea eficiente respecto al sistema que se diseña, por ejemplo, tratar de limitar el uso de periféricos que tienen gran consumo energético o buscar la mayor eficiencia de consumo de batería en sistemas portable, llevando el controlador a estados inactivos en los tiempos que sea posible, así mismo se debe buscar eficiencia en el código de modo que se haga un buen uso de los recursos del hardware tales como el procesamiento y la memoria, disminuyendo la cantidad de variables globales, y hacer buen uso de la pila de memoria, también se recomienda realizar un software modular basado en funciones, en donde el programa principal hace llamados a funciones para el manejo de los datos, este método de programación ayuda a tener una secuencia de programa mucho más organizada, modular y provee un alto factor de calidad de software, pues permite depurar más fácil, hacer reciclaje de código y aumenta la eficiencia del sistema.

Metodología Pahl & Beitz

Un ejercicio que puede ser muy útil en el diseño del hardware, consiste en utilizar la metodología de Pahl & Beitz a partir de la cual se escoge los diferentes tipos de módulos de hardware estableciendo algunos pesos a partir de las especificaciones, el procedimiento es el siguiente:

i. Se genera una tabla con los diferentes módulos que conforman el sistema y los diferentes métodos o arquitecturas que existen a partir de las cuales se pueden cumplir las funciones del módulo, se deja claro que no es compara diferentes referencias de componentes que trabajan del mismo modo, sino de dispositivos que trabajan diferente, pero que hacen lo mismo,

ii. Se generan las rutas posibles a través de los módulos de modo que cada ruta está conformada por una serie de elementos específicos.
Ej.:

R1= Microcontrolador + termocupla + Pantalla monocromática
R2=DSP + RTD + Pantalla monocromática
R3= PLC + termocupla + Pantalla a color
R4= Microcontrolador + RTD + Pantalla Alfanumérica

iii. Se generan una lista de los criterios de evaluación. Esencialmente, la lista de los factores bajo la cual el diseño está siendo juzgado. Las especificaciones de diseño son una fuente de la información para crear esta lista. Esta lista debe ser concisa pero debe referirse a la mayor cantidad de aspectos posible. Los criterios deben ser expresados de manera que sea posible asignar valores numéricos de 0-1, donde 0 es muy pobre y un valor cercano a 1 es excelente y la suma de todos estos criterios debe ser igual a 1.
Ej.:

Costo=0.5
Desempeño=0.3
Versatilidad=0.2

iv. Se evalúan las diferentes rutas generadas de la tabla en cada uno de los aspectos mencionados en el punto anterior, asignando valores a cada uno de modo que dichas valoraciones sumen 10, de modo que 0 quiere decir que tiene un valor muy pobre en ese aspecto de evaluación y 10 que es excelente solo en ese ítem, de este modo se puede comparar cual cumple de mejor manera los requerimientos planteados por el usuario, además de desempeñar las funciones necesarias del sistema.
Ej.:

R1: Costo=6
Desempeño=3
Versatilidad=1
R2: Costo=4
Desempeño=4
Versatilidad=2
R3: Costo=2
Desempeño=4
Versatilidad=4
R4: Costo=7
Desempeño=2
Versatilidad=1

v. Una vez realizadas las valoraciones de las ruta se multiplican dichos valores para el peso de cada uno de los criterios de evaluación, de este

modo es posible observar cuál de las rutas cumple mejor las preferencias que debe tener el sistema a elegir.
Ej.:

R1= 6*(0.5)+3*(0.3)+1*(0.2) = 4.1
R2= 4*(0.5)+4*(0.3)+2*(0.2) = 3.6
R3= 2*(0.5)+4*(0.3)+4*(0.2) = 3
R4= 7*(0.5)+2*(0.3)+1*(0.2) = 4.3

Para el caso del ejemplo se observa que los sistemas con los componentes de la ruta 4 y la ruta 1 cumplen de mejor manera la lista de criterios de evaluación tomada de los requerimientos del sistema.

A partir de este ejercicio es posible realizar una evaluación de todas las posibles formas de realizar un sistema que se ajuste y cumpla con los requerimientos del usuario final.

Parámetros mínimos de hardware y elección de dispositivos.
Una vez que se conocen la mejor configuración de dispositivos, se procede a hacer una valoración del hardware necesario que este en capacidad de correr los programas en ejecución, los tipos de cálculos necesarios, y la cantidad de espacio en memoria que serán utilizados, siempre teniendo muy en cuenta, cumplir las especificaciones de desempeño consignadas por el usuario, por lo general se aconseja utilizar un sobredimensionamiento para poder hacer futuros reajustes o expansiones del sistema, en especial, en sistemas que se sabe que más adelante serán expandidos o mejorados, este sobredimensionamiento se hace en base a la respuesta del usuario en la pregunta de transferencia/actualización de la lista de especificaciones, en donde se conoce que tanto puede ser expandido el sistema y con base en esto, prever la mejor solución de hardware desde el inicio evitando

que más adelante se necesite hacer grandes cambios en el hardware por quedarse corto de capacidad, aumentando así los costos de cualquier mejora.

Se debe buscar conservar la estructura modular del hardware presentada en la etapa anterior, de modo que cada bloque de hardware cumple una función específica, para cada uno de estos módulos se debe generar una tabla de requisitos de hardware, en donde se consignan los parámetros mínimos para que dicho dispositivo este en capacidad de cumplir correctamente su tarea, puesto que cada módulo de hardware tiene características y composiciones especificas a su tarea, no existe una tabla común a partir de la cual se pueda realizar la comparación, de este modo se debe realizar una que contenga la información que sea necesaria.

Tabla 5. Ejemplo de parámetros mínimos para un dispositivo.

Característica	**Valor mínimo**
Velocidad de CPU	10 MIPS
Memoria de programa	32KBytes
Bytes de RAM	2048
Capacidad del canal ADC	12 bits
Numero de pines	40

Elaboración propia

A partir del conocimiento de la operación global que tendrá el sistema y de sus niveles de procesamiento se debe elegir la arquitectura y los dispositivos que cumplan las características necesarias del sistema, dejando un pequeño sobredimensionamiento, teniendo en cuenta también el costo que puedan traer, para el caso de los procesadores se debe tener en cuenta la cantidad y los tipos de datos, así como los tiempos máximos de procesamiento, entre los tipos de procesadores más comunes se encuentran:

Microcontroladores: se utilizan en aplicaciones de control sencillas, que no requieren de muchas prestaciones, poseen buena fiabilidad y se caracterizan por su muy bajo costo.

Procesadores: para aplicaciones que requieren de procesamientos muy grandes de prestaciones medias, en especial para aplicaciones limpias y el manejo de comunicaciones.

FPGA (Field Programmable Gate Array): son utilizadas para aplicaciones que requieren gran procesamiento a altas velocidades y en especial para operaciones en paralelo.

DSP (Digital Signal Processor): se utilizan para aplicaciones con procesamiento de señales que requieren una gran cantidad de operaciones sobre una señal en muy poco tiempo.

PLC (Programmable Logic Controller): este controlador es muy utilizado en la industria, se caracteriza por su gran robustez y está diseñado especialmente para ambientes industriales de gran contaminación electromagnética.

Existen una gran cantidad de plataformas de desarrollos sobre estos procesadores, en este punto el diseñador debe evaluar si realiza desarrollos sobre una plataforma existente en donde utiliza las configuraciones de hardware ya preestablecidas en un sistema de estos o si por el contrario, ya sea por disminución de costos o la necesidad de una aplicación específica, se va a realizar la implementación de un hardware acomodado a su aplicación específica, en caso tal de que sea necesario y se cuente con los recursos, se pueden realizar pruebas físicas sobre dispositivos de los cuales se tenga duda sobre su desempeño, de

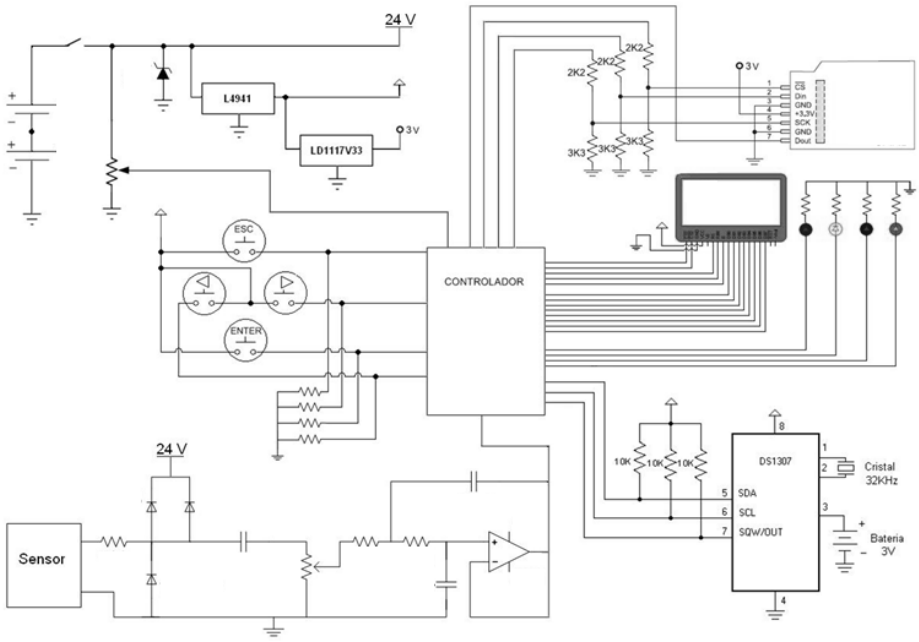
este modo se corrobora que algún módulo propuesto cumple o no con las características necesarias.

Luego de seleccionados los dispositivos específicos que pueden cumplir con las funciones del sistema, se procede a realizar el diseño completo de hardware, teniendo en cuenta las recomendaciones que entrega cada fabricante en sus hojas de datos o datasheet, se debe buscar utilizar las configuraciones más apropiadas y proveer todas las condiciones de alimentación y control necesarias, siendo muy cuidadosos en los elementos que pueden sufrir daños por errores en el diseño, se debe tener cuidado en realizar los cálculos de valores de corriente y voltaje para los elementos que tengan características específicas y no dejar esto a procedimientos de ensayo y error, pues esta práctica puede llevar a quemar el dispositivo.

Puede ocurrir que en esta etapa se detecte algún vacío en el diseño o la falta de algún módulo que realice un proceso necesario para el correcto funcionamiento del sistema, en este caso, se debe retroceder a la etapa anterior y agregarlo, de modo que todo el sistema esté completamente especificado en software y hardware y no encontrar este tipo de inconvenientes más adelante donde el costo de arreglarlo puede ser muy alto, tanto en tiempo, como en costos.

Es importante generar un buen plano eléctrico del sistema, de modo que pueda ser bien interpretado por cualquier persona, de modo que se pueda delegar la tarea de construcción a alguien que no tiene por qué saber el funcionamiento del dispositivo, de igual forma se hace necesario contar con una referencia acerca de la disposición del hardware en caso tal que se presenten problemas con este.

Figura 27. Ejemplo del plano eléctrico de un sistema embebido



Elaboración propia

Una vez ya están elegidos todos los esquemas de hardware y software, se procede a elegir el lenguaje y la herramienta de programación, por ejemplo para los microcontroladores existe el lenguaje Basic, C, Pascal, mientras que en FPGA hay VHDL, Verilog. Y en cuanto a herramientas de programación, existen las propias de los fabricantes, tales como Codewarrior, MPlab, CBuilder, etc, los cuales proveen diferentes herramientas de diseño y librerías de funciones al servicio del usuario, que de cierta manera, pueden ayudar a un desarrollo más fácil y rápido de los programas que necesita el usuario.

Todos estos diseños de programa y configuración de hardware deben ser consignadas en el documento perteneciente a esta etapa, esta documentación puede ser una referencia muy útil en futuros proyectos, debido a que muestra el desarrollo de etapas que pueden ser críticas para otros desarrolladores que no

tienen mucha experiencia en el tema tratado en ese proyecto específico, así mismo este documento es la guía para realizar la codificación en la siguiente etapa, puesto que con un diseño bien detallado, la codificación puede ser fácilmente realizada por alguien que sepa programar, sin necesidad que el desarrollador tenga que cumplir esta tarea, caso muy útil en proyectos que requieran una gran cantidad de código.

Actividades de la etapa:

- Desarrollo de los módulos de software con los algoritmos en pseudocódigo
- Utilizar la metodología de Pahl & Beitz para la evaluación de las diversas posibles composiciones del sistema
- Desarrollo de las tablas con los parámetros mínimos del hardware para su selección
- Elección de una plataforma de desarrollo o elección de componentes para un diseño electrónico
    - Desarrollo del plano electrónico del sistema
- Elección del lenguaje y herramientas de programación

6.4. CODIFICACIÓN Y ANÁLISIS DE RENDIMIENTO

En esta etapa se busca en la medida de lo posible, realizar una validación de los diseños realizados hasta este punto. Para proyectos pequeños y sencillos, esta etapa puede ser obviada, confiando en la experiencia y conocimiento del diseñador, existirán otros casos que bajo la plataforma en la que se trabaja, no sea posible utilizar un programa o modelo virtual en el cual sea posible realizar esta validación o que el costo del mismo, supere sus beneficios, por lo cual también se omitiría.

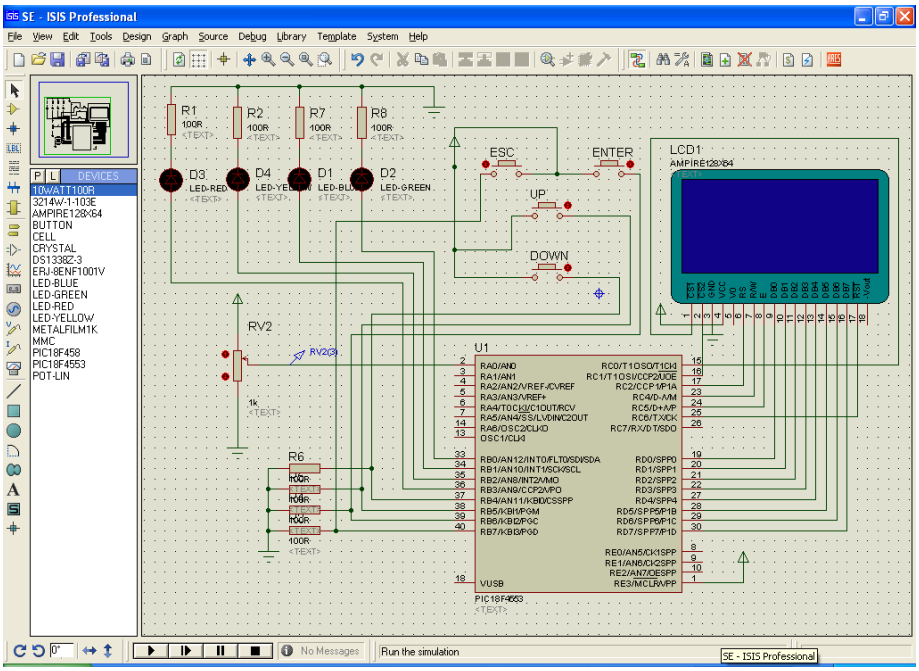
El objetivo de realizar esta etapa es poder confirmar de manera clara que la elección de hardware es correcta, si bien la gran mayoría de componentes

electrónicos se pueden considerar de bajo costo, a veces los procesos de compra pueden llevar un tiempo considerable, y en caso de que no se considere algún componente o se detecte un sub-dimensionamiento de algún otro en una etapa más adelante, se puede incurrir en la pérdida de tiempo y por ende dinero.

En esta etapa también se busca estudiar y analizar que el hardware propuesto en la etapa anterior si sea capaz de cumplir eficientemente con todas las tareas necesarias, para esto se busca la ayuda de software de simulación y herramientas de programación, a partir de las cuales se caractericen los algoritmos desarrollados, se busca entonces constatar que la cantidad de memoria elegida es suficiente para poder almacenar todos los datos esperados y si la capacidad y velocidad de procesamiento es la adecuada y el sistema no va ser muy lento a la hora de procesar todos los datos.

Los pasos a seguir en esta etapa consisten en programar las funciones que más capacidad de procesamiento requieren, tales como operaciones en punto flotante, operaciones con matrices, algoritmos recursivos, módulos de comunicación de datos, etc. la idea es, a modo de ejemplo, simular el comportamiento del sistema ante la ejecución de estas rutinas, tales como tiempos de respuesta, correcta atención a interrupciones, desbordamiento de la pila, problemas con punteros, operación conjunta de módulos o interferencia entre ellos, etc. Cuando se detecte algún tipo de error o malfuncionamiento, se debe retroceder a la etapa anterior y corregir los errores encontrados en el diseño.

Figura 28. Ejemplo de software de simulación hardware



Elaboración propia

En este punto ya se tiene certeza de cuáles son los componentes de hardware del sistema, por lo tanto este es el momento en que se realiza la ubicación y la compra de estos elementos, puesto que normalmente estos trámites son demorados, se puede continuar con la codificación de los diferentes módulos de software, haciendo la traducción de pseudocódigo al lenguaje de programación elegido, para esta función es útil las herramientas de simulación que permiten hacer la depuración del programa, sin necesidad de estar funcionando directamente sobre el hardware.

Una vez que se comprueba que todo hardware y las rutinas de código diseñadas funcionan correctamente en simulación y ya se tienen todos los componentes, se procede a la siguiente etapa.

Actividades de la etapa:

- Evaluación de herramientas de simulación para la plataforma o hardware elegido.
- Decisión de la viabilidad o no de la aplicación de esta etapa, en caso de No, proseguir con la siguiente etapa.
- Selección de los algoritmos o dispositivos con sensibilidad a problemas de ejecución o funcionamiento, en especial los desconocidos por el desarrollador.
- Codificar dichos algoritmos en el lenguaje elegido y distribuir el hardware en simulación tal cual como fue diseñado.
- Simular el comportamiento del sistema.
- Evaluar la respuesta del sistema simulado ante estos programas o configuraciones de hardware.
- Realizar las correcciones necesarias a los diseños en caso de encontrar problemas.

6.5. IMPLEMENTACIÓN DEL HARDWARE

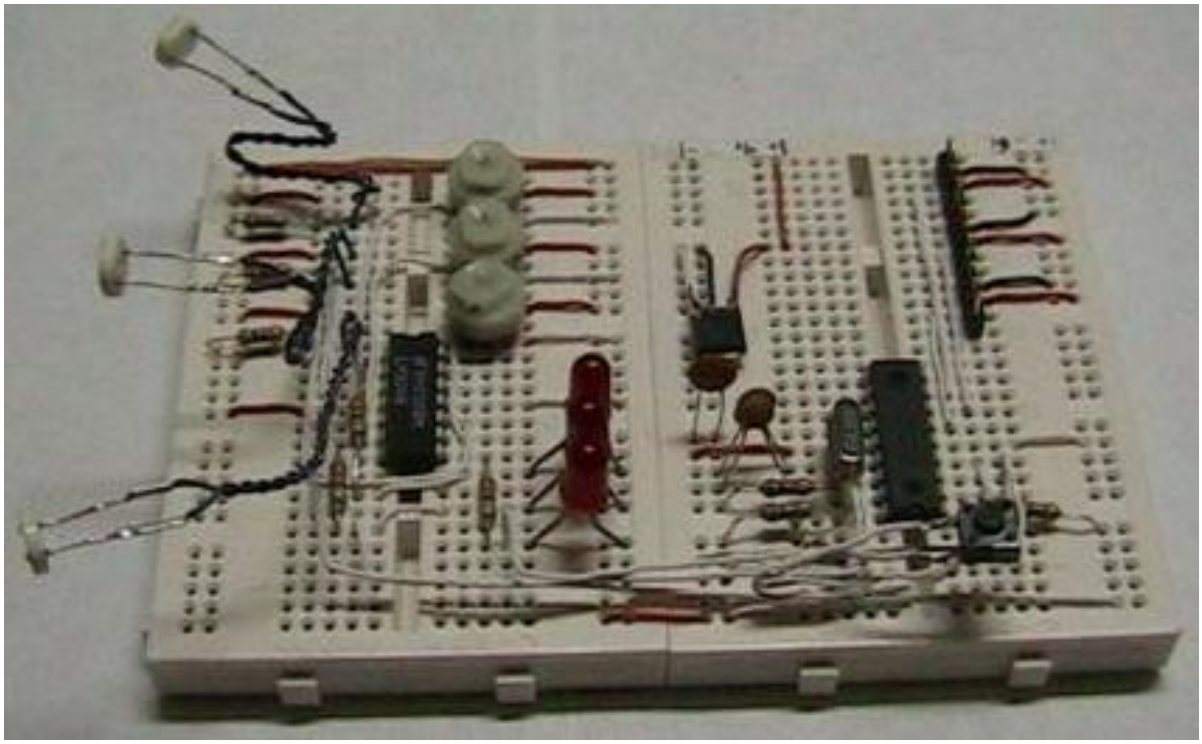
Existen varias posibilidades para la implementación del hardware, todo depende de la complejidad del dispositivo, en caso de que se haya elegido una plataforma de desarrollo comercial que no necesite ningún tipo de periférico, esta etapa se omite completamente.

Para el caso en que se desee realizar un prototipo propio es posible diseñar directamente la tarjeta electrónica para los casos en que se pasó por la etapa anterior de codificación y análisis de rendimiento en la cual se hace la simulación de los circuitos y se está seguro de su funcionamiento, de este modo se previene la situación de generar tarjetas electrónicas que pueden contener errores incurriendo así en la perdida de dinero en la fabricación de elementos.

En caso de no estar seguro del correcto funcionamiento del circuito diseñado es una buena práctica realizar primero algunas pruebas en tarjetas de prototipaje del funcionamiento de los componentes principales, para el caso de circuitos simples, es posible realizar el ensamblaje de los módulos tal como se diseñaron en las etapas anteriores, teniendo cuidado en la correcta manipulación y conexión de todos los elementos, cerciorándose siempre que no queden cables flojos, líneas interrumpidas o malos contactos entre los diferentes módulos del sistema.

Figura 29. Ejemplo tarjeta de desarrollo.

Figura 30. Ejemplo de circuito en prototipaje

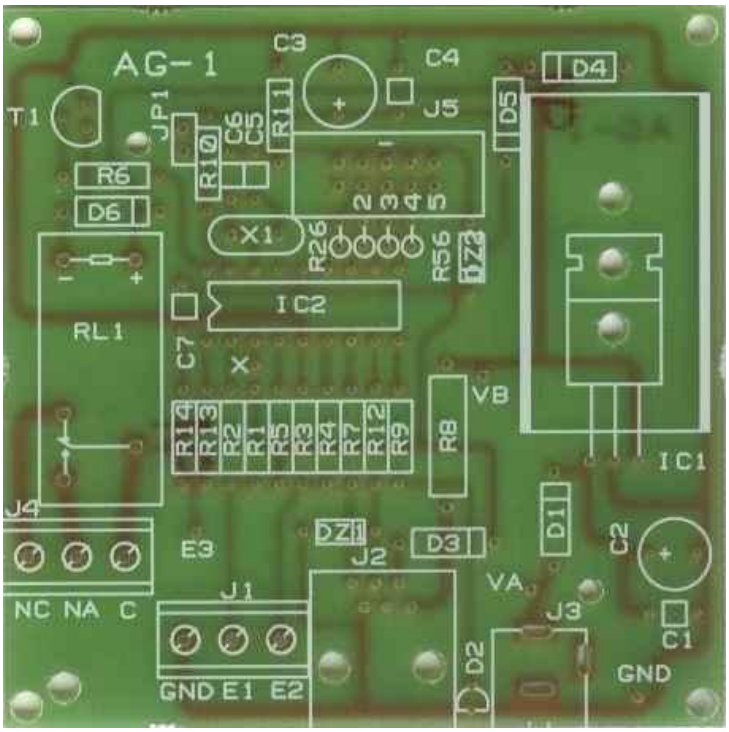


Una vez se construye el hardware, se debe proceder a verificar el funcionamiento de cada módulo, de forma que se tenga certeza de que cada uno está cumpliendo correctamente sus funciones y que todas las señales estén en su funcionamiento normal, ya una vez esté todo el hardware conectado, se procede a programar sobre él diferentes rutinas de pruebas para ensayar el correcto funcionamiento de todos los periféricos del sistema, esto es probar la lectura de los botones, la digitalización de valores análogos, pruebas de visualización, etc. esta evaluación es de gran importancia debido a que se aminora la cantidad de tiempo de búsqueda de la causa de problemas cuando el sistema no funciona, descartando inicialmente, los problemas de hardware.

En este punto es muy conveniente dedicar bastante tiempo en la parte de acondicionamiento de las señales y la adquisición de datos, pues como se mencionó anteriormente, de esto depende la precisión que tenga el sistema, es un buen ejercicio comparar los valores medidos con otros equipos de medición para tener la certeza que en las etapas posteriores se trabajara con los valores correctos.

Una vez se tiene plena seguridad de que el hardware funciona correctamente y se han corregido las dificultades encontradas, será posible fabricar las tarjetas del prototipo final, en las cuales se realizara la programación completa de todo el sistema.

Figura 31. Ejemplo tarjeta circuito impreso.



Una vez las tarjetas son manufacturadas y el sistema ensamblado, será posible continuar a la par, con el desarrollo del sistema, el proceso de fabricación de la carcasa y demás elementos externos del sistema.

El documento de esta etapa debe tener consignado la descripción de las pruebas y montajes que se hicieron, así como los problemas encontrados durante estos ensayos, los cuales pueden ser propios de la arquitectura como tal o de la falta de conocimiento de situaciones específicas que requieren algunos elementos, este documento es útil para cuando vuelvan ocurrir errores más adelante, ya exista una solución escrita a dicho problema.

Actividades de la etapa:

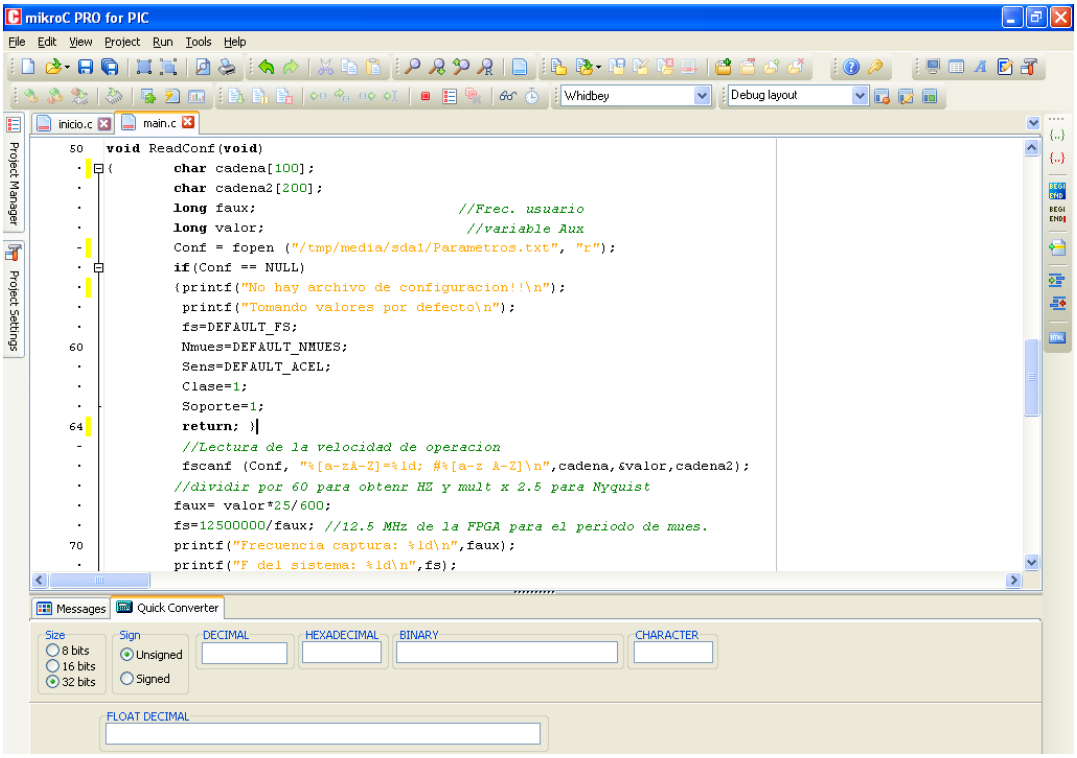
- En caso de haber elegido una plataforma de desarrollo que no requiere más componentes se puede pasar a la siguiente etapa.
- Si se necesita complementar la tarjeta, en esta etapa se construyen los diversos módulos y se prueban, de modo que en etapas posteriores no sea necesario solucionar problemas referentes a funcionamiento de hardware
- Para el caso de diseño específico se recomienda realizar prototipaje de los módulos de hardware de los que se tenga duda en su funcionamiento, de modo que se descarte errores de diseño antes de diseñar el circuito impreso.
- Diseño y fabricación del circuito impreso del sistema
- Desarrollo de la carcasa del sistema, proceso de fabricación de la carcasa y demás elementos externos del sistema.
- Una vez armado el hardware se debe comprobar el correcto funcionamiento de cada uno de los módulos
- Solo hasta que todo el hardware del sistema esté totalmente funcional se debe pasar a la siguiente etapa, esto para descartar errores de hardware en problemas encontrados en la programación del software

6.6. PROGRAMACIÓN DEL SISTEMA.

Para esta etapa ya se tiene certeza que todo el hardware del sistema está conectado y operativo, además de la tercera etapa, se tiene ya codificadas unas rutinas de códigos que cumplen funciones del sistema. En esta etapa se hace una especie de integración de todas esas rutinas en un programa principal que las administra a todas, además se hacen los ajustes necesarios para integrar las rutinas que manejan los periféricos del sistema, esto es la configuración de registros de control, los ajustes de tiempo y la simultaneidad de funciones, por ejemplo hay periféricos que no pueden operar mientras haya interrupciones habilitadas y viceversa. Lo primero que se hace es crear la máquina de estados principal o sistema de control a partir del cual se dará control a la ejecución del programa, posteriormente se empiezan a agregar las funciones ya codificadas y se hacen pruebas de funcionamiento, junto con las anteriormente ingresadas y así sucesivamente hasta tener todos los módulos funcionales en el sistema.

En caso que el sistema embebido tenga una interfaz gráfica e interacción con el usuario, se inicia entonces la programación y codificaciones de las funcionalidades del sistema, siempre teniendo en cuenta que se debe brindar la mayor facilidad para que el usuario opere el sistema, esta etapa es de las que mayor tiempo de programación lleva, pues se debe buscar que no se generen errores en su ejecución, debido a que esta es la parte que manipulara el usuario, algún tipo de malfuncionamiento o errores en la parte de visualización, le pueden hacer pensar que el sistema embebido no funciona correctamente a pesar que toda la parte de procesamiento de las señales sea correcta. De igual forma la implementación de visualizaciones e interfaces humanas con los dispositivos son grandes extensiones de programa debido a la gran cantidad de combinaciones posibles que puede ingresar el usuario mientras manipula el sistema, así mismo se busca también realizar la mayor cantidad de pruebas de manipulación posibles en esta etapa del sistema.

Figura 32. Ejemplo entorno de programación.



El documento de esta etapa muestra el código fuente del programa principal, en donde se debe notar claramente la ejecución y a partir del cual se llaman las funciones descritas en la etapa de codificación y análisis del sistema, con toda la documentación del caso y las notas que sean necesarias.

Actividades de la etapa:

- Programación del módulo principal del programa que llamará los diferentes módulos de software.
- Programación de los módulos de software diseñados en etapas anteriores.
- Realizar pruebas de funcionalidad de los diferentes módulos del sistema y su correcto funcionamiento sobre el hardware.

6.7. PRUEBAS, DEPURACIÓN Y FINALIZACIÓN

En esta etapa se busca realizar la mayor cantidad de pruebas posible del sistema, en caso de que se adquieran datos, se deben comparar con las Mediciones capturadas en otros dispositivos de adquisición, esta evaluación es realizada por el desarrollador del sistema, teniendo en cuenta los test propuestos por el usuario y los ítems especificados en la toma de requerimientos, también se puede hacer uso de las herramientas de depuración que brindan los software de desarrollo para asegurarse que no hayan problemas de flujo de datos o posibles errores de desbordamiento en la pila de memoria, también es importante realizar pruebas utilizando valores de pruebas cercanos a los límites de operación para comprobar que el sistema no se vuelve inestable.

Hay que tener en cuenta que esta etapa demanda mucho tiempo, pues es necesario que se tenga plena seguridad que todo el sistema cumpla con todos los detalles específicos, lo que hace que el tiempo de evaluación sea considerable, durante la ejecución de estas pruebas el desarrollador del sistema embebido debe realizar los ajuste y calibraciones necesarios, aprovechando también, para obtener la hoja técnica característica del sistema final en donde están consignadas los valores de operación reales del producto desarrollado.

Por último en esta fase se dan los ajustes finales del sistema, se brinda la protección mecánica de los circuitos, tales como carcasas, aislamientos, sellados, etc. desarrollados a partir de los requerimientos del usuario.

En el documento de esta etapa se hace un recuento de los bugs o errores encontrados, esto sirve para ir creando una base de datos de los errores más comunes que pueden ocurrir en la ejecución del sistema, para que cada nuevo sistema producido tenga menos probabilidad de tener estas fallas.

Una vez el sistema cumple para el desarrollador con todas las especificaciones se procede a realizar la documentación del sistema tales como la hoja técnica de datos que reúna la información relevante tal como los parámetros de operación, rangos de medición, etc. así como los manuales de mantenimiento, manual de usuario, manual de funcionamiento y demás documentos que se hayan acordado en la fase inicial de especificaciones tales como planos, programas y demás.

Actividades de la etapa:

- Realizar pruebas en el sistema en función de la lista de requerimientos, buscando que el sistema si cumpla todos los numerales de la lista.
- Realizar las pruebas que sugiere el usuario respecto a cómo probar el equipo, presente en la encuesta realizada al usuario en la primera etapa.
- Instalación de la carcasa y demás elementos del aspecto físico del sistema final.
- Evaluación detallada del sistema y pruebas funcionales.

6.8. EVALUACIÓN

Esta es la etapa final del desarrollo en donde un tercero realiza la evaluación del sistema con base en el documento de especificaciones del usuario, se realizan pruebas de campo y de igual forma se hace comparaciones con otros equipos de adquisición. Una vez realizadas dichas pruebas, el tercero suministra su evaluación del producto y el concepto de si cumple o no con las características enunciadas en el documento de especificación realizado al inicio del proyecto.

Cuando se pretende que el producto sea producido en serie, se debe dejar un tiempo de evaluación pertinente por parte del prototipo para descartar posibles problemas de calidad en los componentes, ya sea que se vuelvan inestables en el tiempo, o el proveedor tenga en sus planes sacarlos del mercado, una vez se

tenga la información de que no van a existir problemas en los insumos, se pueden empezar a producir en serie el producto desarrollado.

A partir de la evaluación final se determina si es necesario corregir algún aspecto del sistema que haya quedado faltando, ya cuando se llegue a un común acuerdo que todo funciona perfectamente, se debe igualmente firmar un acta de finalización de proyecto donde se manifiesta que se cumplieron los objetivos del desarrollo, además se pueden incluir demás aspectos acordados anteriormente como lo son la garantía, préstamo de servicio de mantenimiento y demás aspectos que se hayan acordado respecto al final del proyecto.

Actividades de la etapa:

- Elección en conjunto con el usuario de un tercero que evalúe el sistema a partir del documento de especificaciones.
- En caso de ser necesarias correcciones, se deben realizar, mientras estén en los compromisos firmados al inicio del proyecto.
- Firma del acta de finalización del proyecto.

7. CONSTRUCCIÓN

A partir del protocolo expuesto en el capítulo anterior se procede a iniciar el desarrollo del sistema de medición de vibraciones siguiendo paso a paso las etapas propuestas. Como etapa inicial, se hizo una encuesta al usuario.

Se recogieron una serie de requerimientos en los que se reúnen las características que debe tener el sistema, a partir de estas especificaciones se continúa con el diseño general en donde se idealiza la composición del sistema a partir del diagrama de bloques, luego se realiza el diseño en detalle de cada uno de esos bloques, teniendo entonces todos los esquemas se realiza una simulación del desempeño y funcionalidad del todo el sistema, una vez se obtuvieron buenos resultados, se procede a la fabricación del circuito y a la programación de todo el sistema.

A continuación se muestra el desarrollo de cada etapa llevada a cabo durante el proyecto y en los anexos se encuentran los documentos específicos de cada etapa.

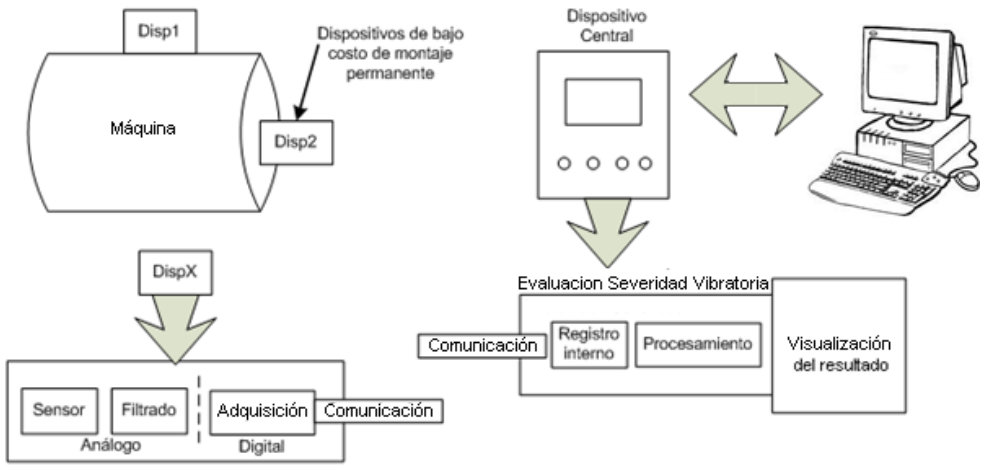
7.1. ESPECIFICACIONES

Como etapa inicial del proyecto, se recogieron una serie de características a partir de las cuales se desarrolla el producto, a continuación se muestra un resumen general de las características que tendrá el producto tomado a partir de la encuesta realizada, en el ANEXO C se encuentra la captura de la encuesta realizada al usuario y en el ANEXO D se muestre el desarrollo del documento de especificaciones del sistema, a continuación se muestran los aspectos más relevantes del sistema.

Se desea desarrollar un sistema que evalúe de la severidad vibratoria por medio de las normas ISO 2372 e ISO 10816-3. Como componente nuevo, se requiere que el sistema esté basado en puntos de medición de bajo costo, los cuales puedan ser incrustados permanentemente en la máquina a medir, cada punto de medición contiene el sistema de acondicionamiento y digitalización de la señal, por otro lado, como sistema portable, se debe desarrollar un módulo central que se comunique con los puntos de medición, el cual realiza el procesamiento de los datos, almacenamiento y visualización de los resultados del sistema.

El dispositivo central debe contener un sistema de botones, una pantalla de indicación y cuatro luces de colores diferentes, a través de las cuales se indica en cuál de los estados de severidad vibratoria se encuentra la máquina, según la norma elegida. Se requiere que el sistema sea de fácil operación, de modo que una persona sin ningún conocimiento técnico pueda llevar a cabo la medición.

Figura 33. Sistema de evaluación de la severidad vibratoria a desarrollar



Una vez terminadas las Mediciones, se pueden descargar los resultados al PC a través de un cable USB, de modo que aparezcan, como si fuere una memoria, en un archivo de texto que pueda ser importado a una plantilla de Excel.

7.2. CONCEPCIÓN DEL SISTEMA

En esta etapa se busca mostrar de manera jerárquica la composición del sistema para posteriormente hacer un diseño detallado, a partir de las especificaciones de la etapa anterior. En el ANEXO E se encuentra el documento realizado para esta etapa donde se encuentra todo el desarrollo completo de los diagramas, a continuación se hace una descripción general de los diseños conceptuales realizados.

7.2.1. Normas ISO 2372 e ISO 10816-3

Estos estándares internacionales definen bases y reglas específicas a emplear en la evaluación de la vibración mecánica de máquinas rotativas, cada norma está realizada específicamente para máquinas rotativas que trabajan en ciertos rangos de rotación, y con base en la potencia, las clasifican en diversas clases o grupos, ambas normas utilizan como parámetro de decisión el valor RMS de la velocidad vibratoria, proporcionando unas ecuaciones que relacionan este valor con la aceleración, velocidad y desplazamiento de la vibración, estas relaciones son mostradas en mayor detalle en el ANEXO E.

La norma ISO 2372 es del año 1974, mientras que la norma ISO 10816-3 es del año 1998, si bien se considera una actualización de la norma anterior, ambas son ampliamente utilizadas, a continuación en la Tabla 6 se muestra la comparación entre ambas normas en donde se puede observar las máquinas que cobijan cada norma y las clasificaciones que realiza cada uno a partir de la potencia y características de la máquina.

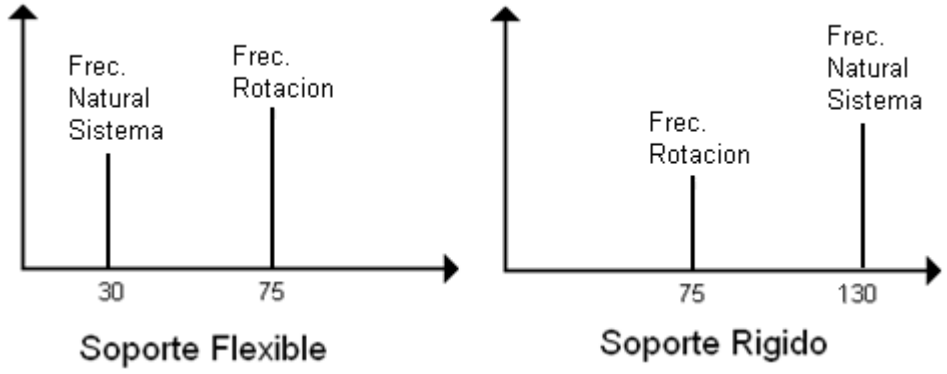
Tabla 6. Comparación normas ISO 2372 e ISO 10816-3

Norma	ISO 2372	ISO 10816-3
Rango de revoluciones	60 rpm – 1200 rpm	120rpm – 15000rpm
Frecuencia de rotación	1 – 20 Hz	2 - 250Hz
Frecuencia instrumento	10 - 1000 Hz	10 - 1000 Hz
Clase I	<15kW	>300kW
Clase II	15-75kW	>15Kw-300kW
Clase III	>75kW, Sop. rígido	>15Kw, motor separado
Clase IV	>75kW, Sop. flexible	>15Kw, motor integrado

Elaboración propia

Los soportes de una máquina, se clasifican en dos tipos, soportes flexibles y soportes rígidos, en los soportes flexibles, la frecuencia fundamental del sistema soporte-máquina es más baja que su frecuencia principal de excitación (en la mayoría de los casos es la frecuencia de rotación), para un soporte rígido, la frecuencia fundamental del sistema soporte-máquina es mayor que su frecuencia principal de excitación.

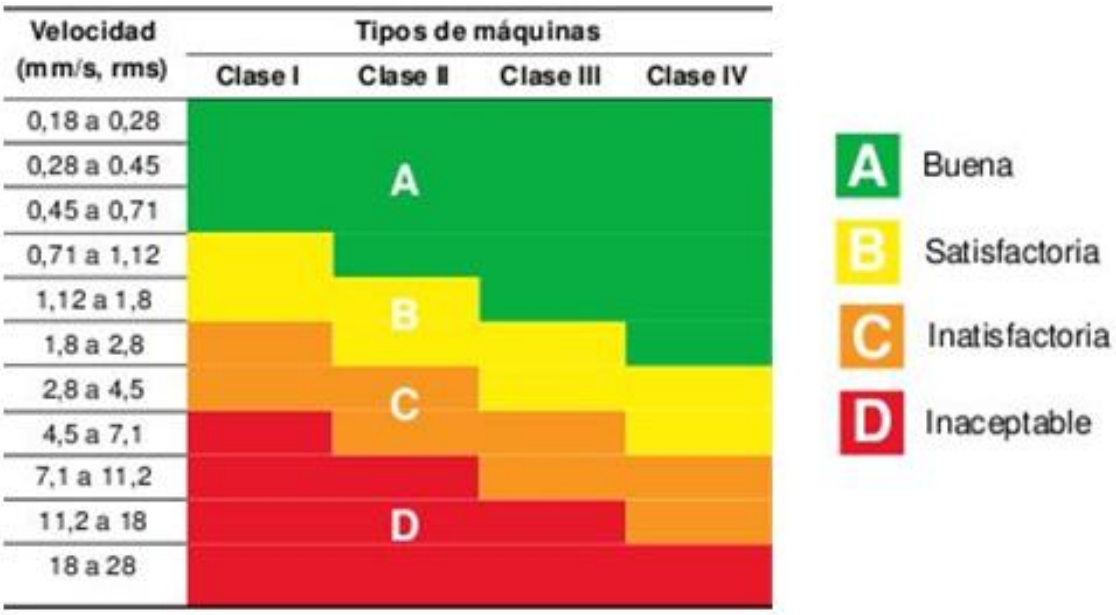
Figura 34. Tipos de soporte en una máquina.



Saavedra, Bases del mantenimiento predictivo y del diagnóstico de fallas en máquinas rotatorias, 2000.

En las Tablas 7 y 8 se muestran las clasificaciones realizadas por cada norma para establecer a partir del valor RMS de la velocidad vibratoria, el nivel de severidad vibratoria en una máquina rotativa.

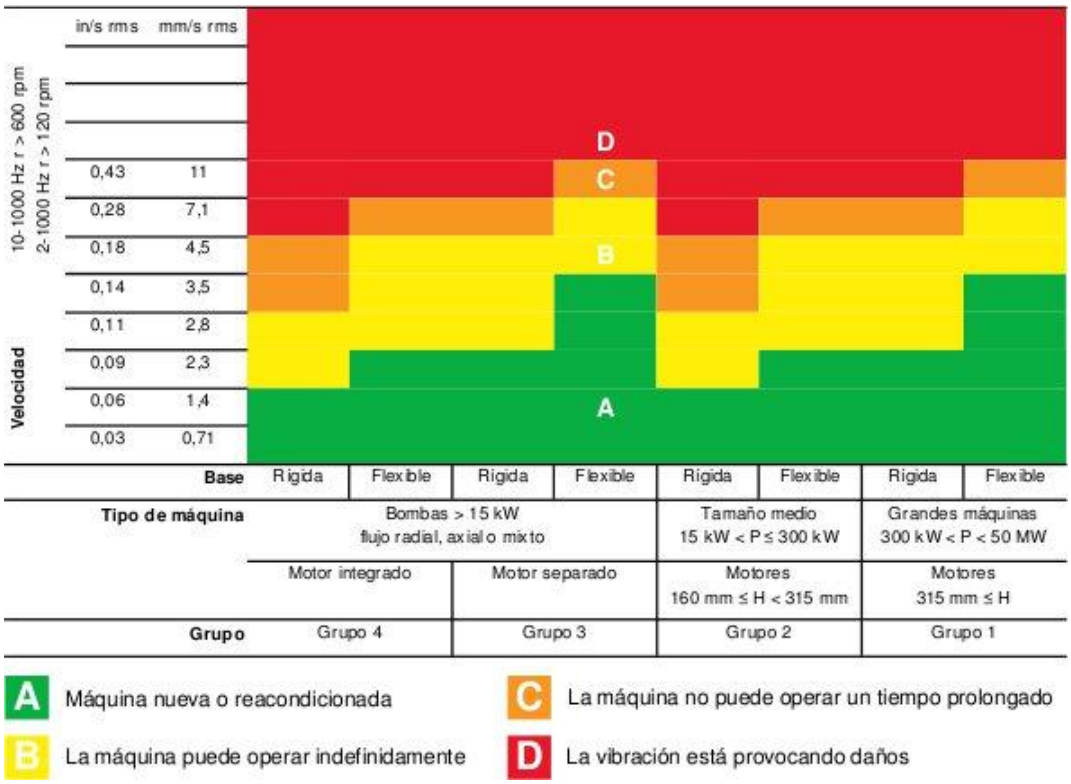
Tabla 7. Clasificación de la severidad vibratoria según norma ISO 2372



Saavedra, Bases del mantenimiento predictivo y del diagnóstico de fallas en máquinas rotatorias, 2000.

Hay que tener en cuenta que las vibraciones medidas en la superficie de la máquina solo pueden proveer una indicación de los esfuerzos vibratorios dentro de la máquina, y no necesariamente reflejaran los esfuerzos reales de las partes críticas, ni tampoco aseguraran que no ocurran excesivos esfuerzos vibratorios, debidos por ejemplo, a resonancias locales. En particular, las vibraciones torsionales de las partes rotatorias no siempre generarán vibraciones medibles en la superficie de la máquina. [ISO 1974, 1998]

Tabla 8. Clasificación de la severidad vibratoria según norma ISO 10816-3

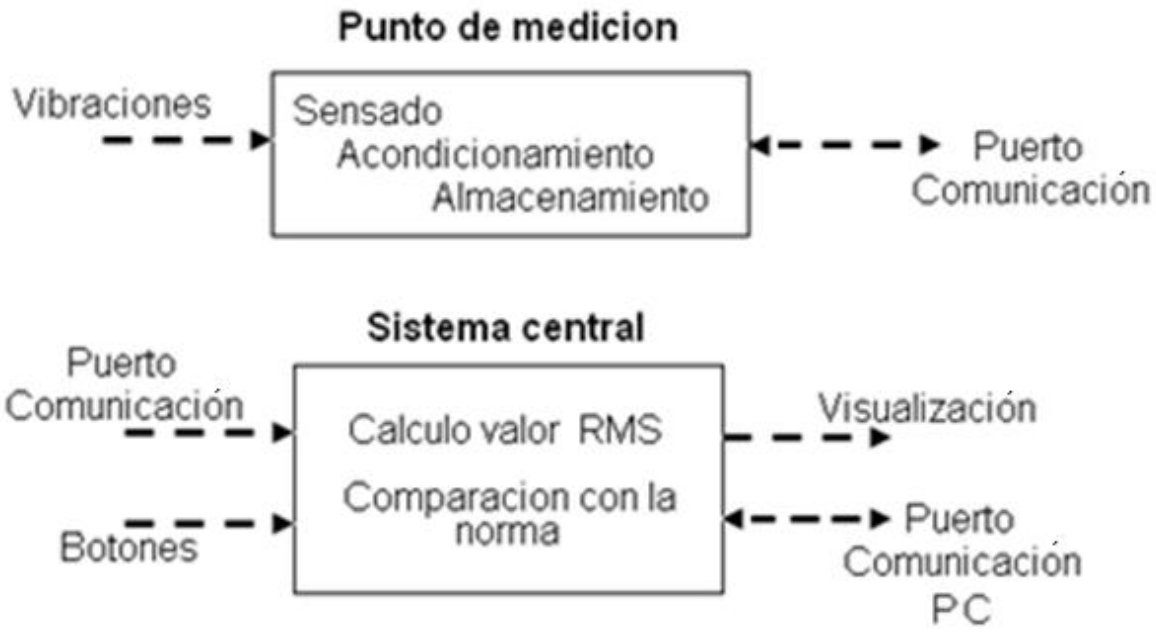


Saavedra, Bases del mantenimiento predictivo y del diagnóstico de fallas en máquinas rotatorias, 2000.

7.2.2. Diagrama de caja negra

A partir de las especificaciones del usuario se procede a realizar el diagrama de caja negra del sistema el cual es mostrado en la Figura 35.

Figura 35. Diagrama de caja negra del sistema

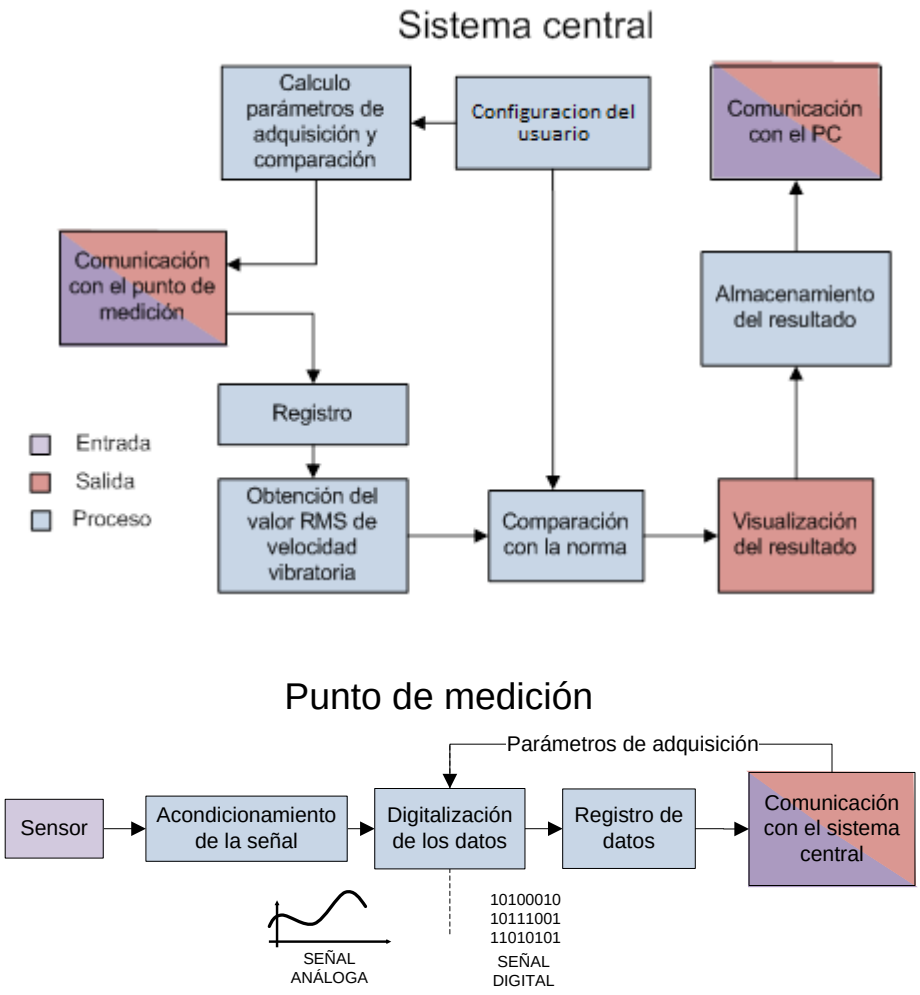


Elaboración propia

7.2.3. Flujo de la información

Una vez se conoce la norma y la respuesta que se le debe entregar al usuario al terminar la medición, se realiza el diagrama de flujo de la información en donde se busca mostrar de forma modular, cuales son los diferentes procesos o transformaciones que sufre la información en el sistema, a partir de las especificaciones, se tiene que el sistema está compuesto por dos módulos diferentes: un sistema llamado punto de medición donde se realiza la adquisición de datos y un sistema central donde se hace el procesamiento de la información según la norma, con base en esto, se generan un flujo para cada módulo, a continuación se muestra el diagrama.

Figura 36. Flujo de información del sistema



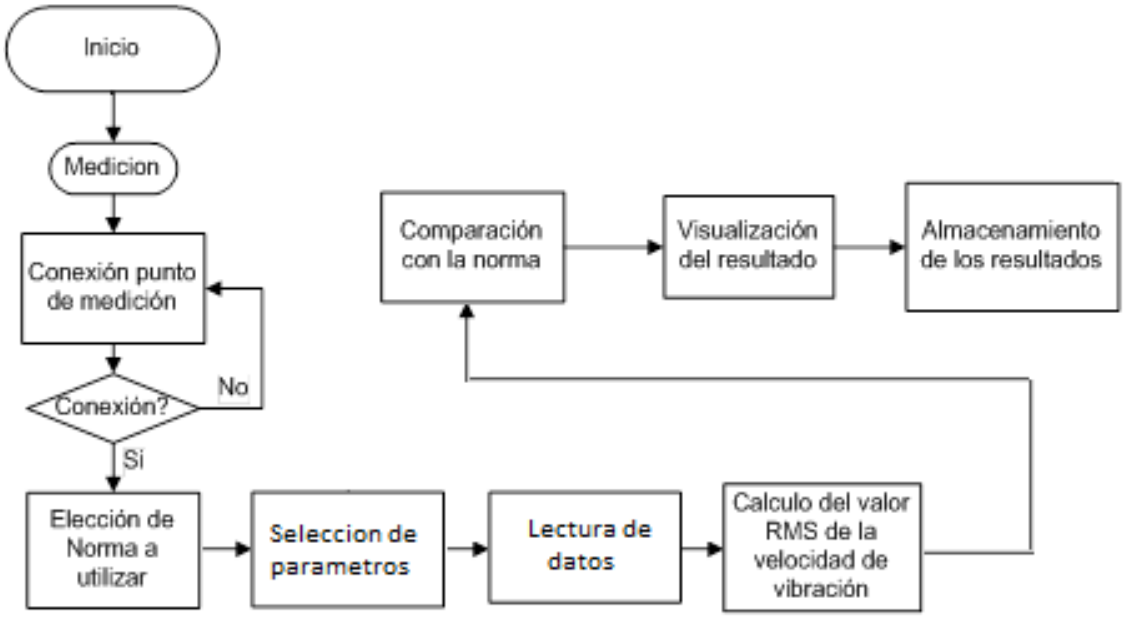
La descripción detallada de este diagrama y las transformaciones que sufre la información durante cada una de las etapas de este diagrama se encuentran en el ANEXO E. Concepción del sistema.

7.2.4. Software

Del documento de especificaciones consignado en el ANEXO D y del diagrama del flujo de información, se hace un diagrama que contiene las funciones principales de software a partir de las cuales será posible cumplir las funciones especificadas,

se designan tres menús principales, los cuales requieren de una serie de funciones para llevar a cabo sus tareas, en las Figuras 37 y 38 se muestran la forma general de las funciones necesarias del sistema central y el punto de medición, la descripción y desarrollo de cada una de estas funciones se encuentra en el ANEXO E. Concepción del sistema.

Figura 37. Diagrama general de software del sistema central

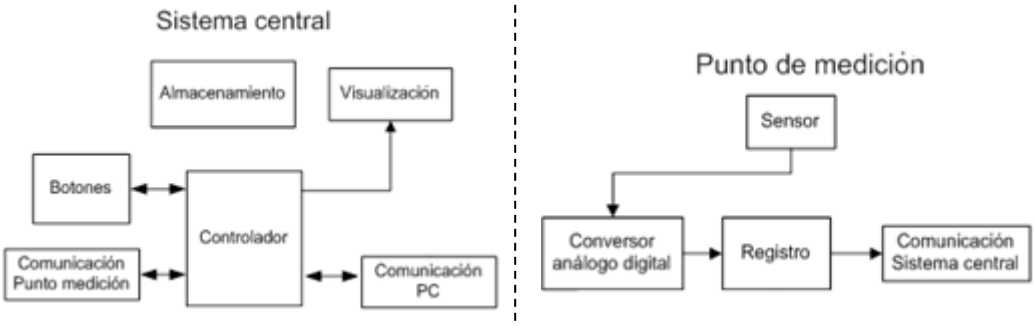


Elaboración propia

7.2.5. Hardware

Del documento de especificaciones consignado en el ANEXO D y del diagrama del flujo de información, se hace un diagrama de hardware a partir del cual será posible cumplir las funciones especificadas, el diagrama es el mostrado en la Figura 39, en este se consignan los diferentes módulos de hardware necesarios para procesar, comunicar y almacenar la información del sistema.

Figura 38. Diagrama de bloques de hardware



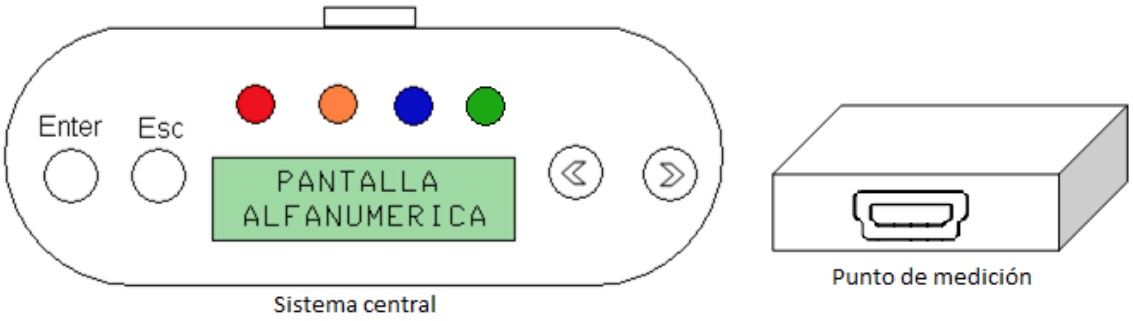
Elaboración propia

Con este hardware se puede cumplir con todas las especificaciones, en el ANEXO E. Concepción del sistema, se encuentran la descripción y los diagramas de bloques específicos para cada etapa.

7.2.6. Carcasa del sistema

En esta etapa del protocolo se busca generar la idea de la forma física de la carcasa del sistema, puesto que el usuario ya posee una idea de esta y además posee una forma agradable y bien distribuida, no es necesario definir por ahora más aspectos relacionados con la carcasa, en la Figura 39 es mostrado el diagrama inicial de la carcasa, la cual puede variar en algunos aspectos según las necesidades y requerimientos para su fabricación una vez se levanten las características finales con base en el hardware construido de los dos sistemas. El cual, al momento de ser diseñado, debe buscar corresponder a esta forma presentada.

Figura 39. Forma de la carcasa del sistema



7.2.7. Cronograma de actividades

A partir de todos los análisis para cada etapa de hardware y software, se realiza una lista de actividades a seguir para el correcto avance del proyecto, se da un tiempo estimado de la duración de actividades considerando el tiempo promedio de desarrollo sin muchos inconvenientes, además de contar con la disposición de los componentes necesarios, a partir de todos estos elementos y del tiempo de desarrollo considerado para cada etapa, se genera el cronograma de actividades mostrado en la Tabla 9.

Tabla 9. Cronograma de actividades del proyecto

Actividad	Tiempo (semanas)												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Diseño Detallado													
Análisis de rendimiento													
Pruebas de módulos de hardware													
Pruebas de adquisición sensores													
Implementación del hardware													
Programación del sistema													
Elaboración de carcasa													
Pruebas y depuración de sistema													
Evaluación del sistema													

7.3. DISEÑO DETALLADO

En esta etapa se busca diseñar al detalle todos los componentes de hardware y software con base en el diseño bloques general del sistema (Figuras 37 y 38), a continuación se inicia con la descripción del diseño del software y hardware, el documento completo con toda la información respectiva de esta etapa de desarrollo, se encuentra en el ANEXO F de este informe.

7.3.1. Diseño del software

En esta fase del diseño se generan los pseudocódigos de las principales funciones del sistema, se inicia mostrando el modo en que se hace la visualización de mensajes y el ingreso de datos, la cual estará presente en gran parte de todo el programa, luego se muestra un ejemplo de cómo es la máquina de estados que regirá la ejecución de las diferentes rutinas y funciones del sistema.

En el ANEXO F se encuentra la descripción de cada función, al final de ese anexo se encuentra la lista completa de todos los algoritmos en pseudocódigo, cada uno debidamente numerado y titulado para evitar confusiones a la hora de hacer referencia, en ellos se describe brevemente la forma de implementación de cada función donde se muestran las operaciones y las decisiones que se toman a partir de los datos de entrada de cada segmento de código, los algoritmos que allí se encuentran son:

- Adquisición de datos
- Máquina de estados del sistema
- Conexión punto de medición
- Elección de la norma a utilizar
- Enviar parámetros de adquisición
- Registrar los datos recibidos
- Cálculo del valor RMS de la velocidad vibratoria

- Comparación con la norma
- Visualización del resultado
- Almacenamiento de los resultados
- Conexión con el PC
- Registro y cálculo de parámetros de adquisición y comparación
- Almacenamiento de tabla de parámetros
- Enviar Mediciones almacenadas
- Borrar Mediciones almacenadas en el sistema
- Configurar fecha y hora
- Conexión con sistema central
- Adquisición de datos
- Registro de datos

A partir del conocimiento de todas estas funciones, se inicia entonces con el diseño detallado de la composición de hardware del sistema.

7.3.2. Diseño del hardware

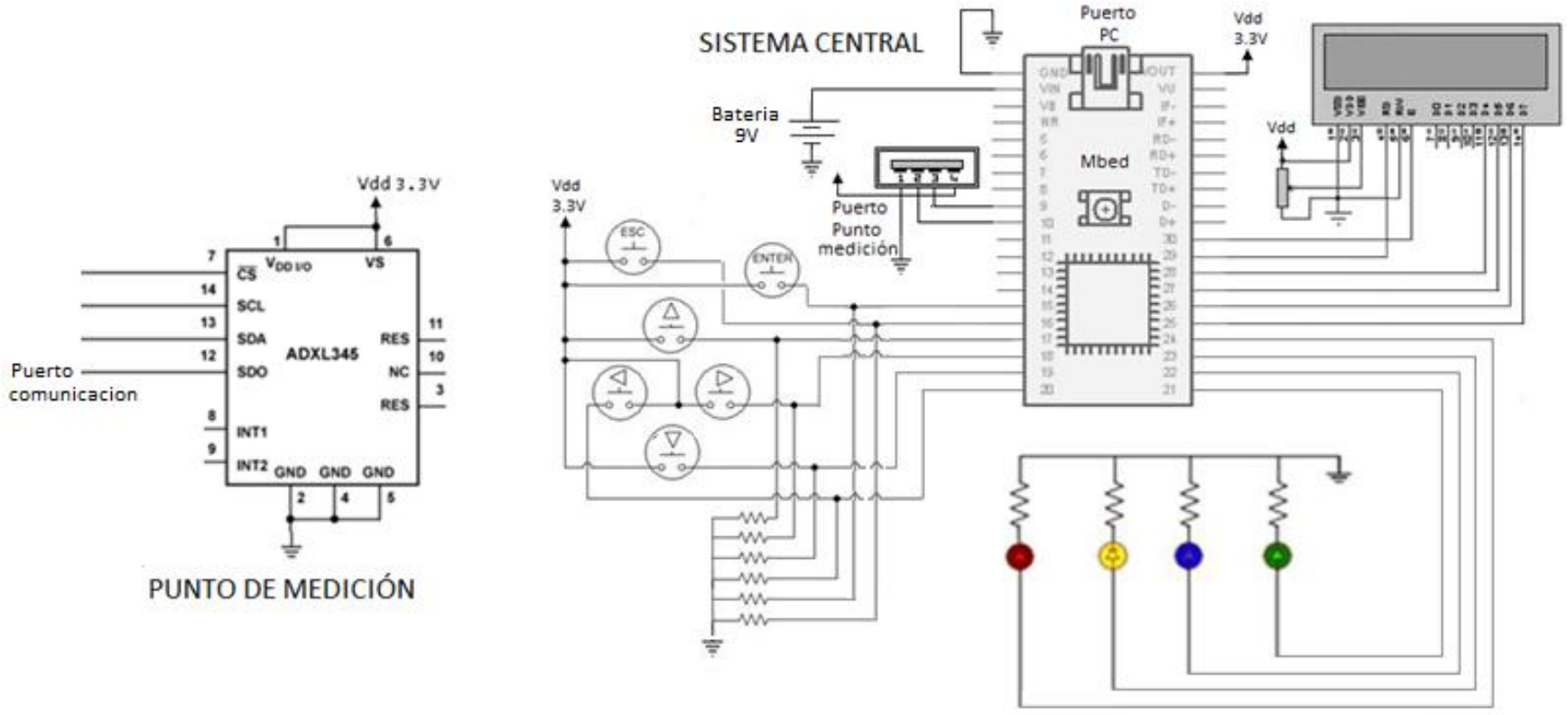
Siguiendo el orden propuesto por el protocolo de desarrollo, se inicia el desarrollo de hardware utilizando la metodología de Pahl & Beltz en donde se ponen a prueba diferentes tipos de tecnologías que están en capacidad de cumplir con las funciones del sistema a desarrollar, como parámetros de evaluación para este proyecto se eligieron los aspectos de costo, tiempo de desarrollo y desempeño, entre las tecnologías comparadas se encontraban los DSP, las FPGA, módulos de memoria Flash, y algunos protocolos de comunicación, luego de realizar la evaluación, se encontró que la combinación que mejor cumplía con dichos parámetros fue:

Controlador: Microcontrolador
Almacenamiento de datos: Memoria Flash
Registro de datos: Memoria RAM

Protocolo comunicaciones: SPI

En el Anexo F se encuentra todo el procedimiento detallado de este paso del protocolo, posterior a esto, fueron realizadas una serie de tablas que reunían los parámetros mínimos de elección de cada módulo de hardware, así como una serie de tablas comparativas de diversos productos y estándares que estaban en capacidad de cumplir con los requerimientos, una vez se realizó la elección de todos los componentes del sistema, se elaboró el plano electrónico del sistema, el cual es mostrado en la Figura 40.

Figura 40. Diagrama detallado del hardware del sistema.



Elección del lenguaje y herramientas de programación.

En este caso el sistema Mbed posee una plataforma de desarrollo basada en web, la cual posee una gran cantidad de librerías y ejemplos que son de gran ayuda para la utilización de todas sus funcionalidades, el lenguaje de programación es C++.

Elección de la carcasa del sistema.

Con el fin de disminuir los costos que conlleva el diseño y fabricación de una carcasa personalizada, se opta por buscar en el mercado diferentes tipos de carcasas comerciales de modo que se acomode a la aplicación que se quiere desarrollar, luego de una intensa búsqueda, evaluación de costos, ventajas y desventajas de cada una, fueron elegidas dos carcasas que se muestran a continuación.

Carcasa del sistema central

Figura 41. Carcasa del sistema central

Caja de plástico 152x83x33 mm de tres cuerpos y ventana para LCD de 2 pulgadas

Carcasa del punto de medición

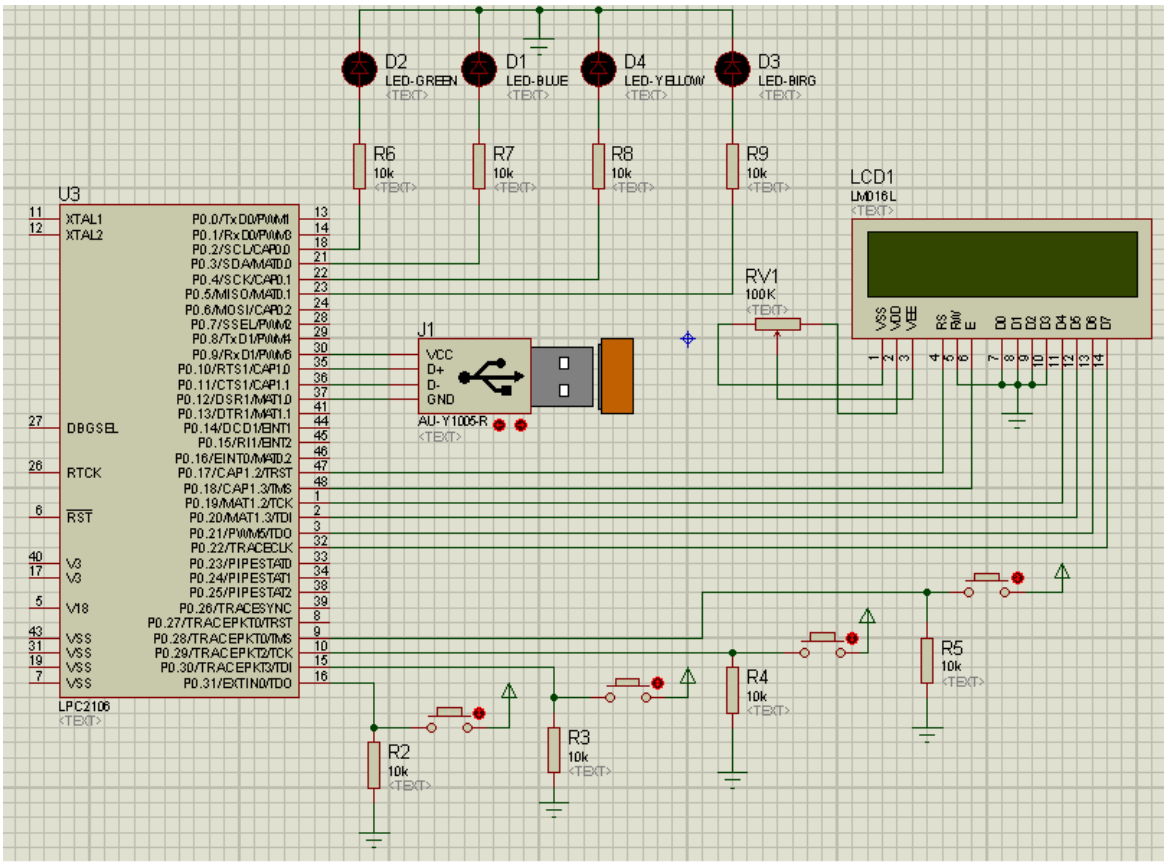
Figura 42. Carcasa del punto de medición

Caja de aluminio con tapas de tamaño 53x44x12 mm

7.4. CODIFICACIÓN Y ANÁLISIS DE RENDIMIENTO

En esta etapa se realizaron una serie de pruebas virtuales en el entorno de simulación del programa *Proteus* con el fin de corroborar el correcto funcionamiento de la configuración de hardware diseñada, para esto se generó el esquemático mostrado en la Figura 43, las pruebas realizadas mostraron resultados satisfactorios, los detalles del desarrollo de esta etapa se encuentran en el ANEXO G. Codificación y análisis de rendimiento.

Figura 43. Circuito diseñado para el Sistema central



7.5. IMPLEMENTACIÓN DEL HARDWARE

En esta etapa se generó todo el hardware del sistema, inicialmente se hicieron una serie de pruebas de hardware en tarjetas de prototipaje para comprobar el correcto funcionamiento del sistema, posterior a esto, se generaron los planos electrónicos y el diseño del circuito impreso para cada uno de los dos sistemas los cuales son mostrados en las Figuras 44, 45 y 46. En el ANEXO H. Implementación del hardware se muestran más detalles de este proceso.

Figura 44. Prototipaje en board

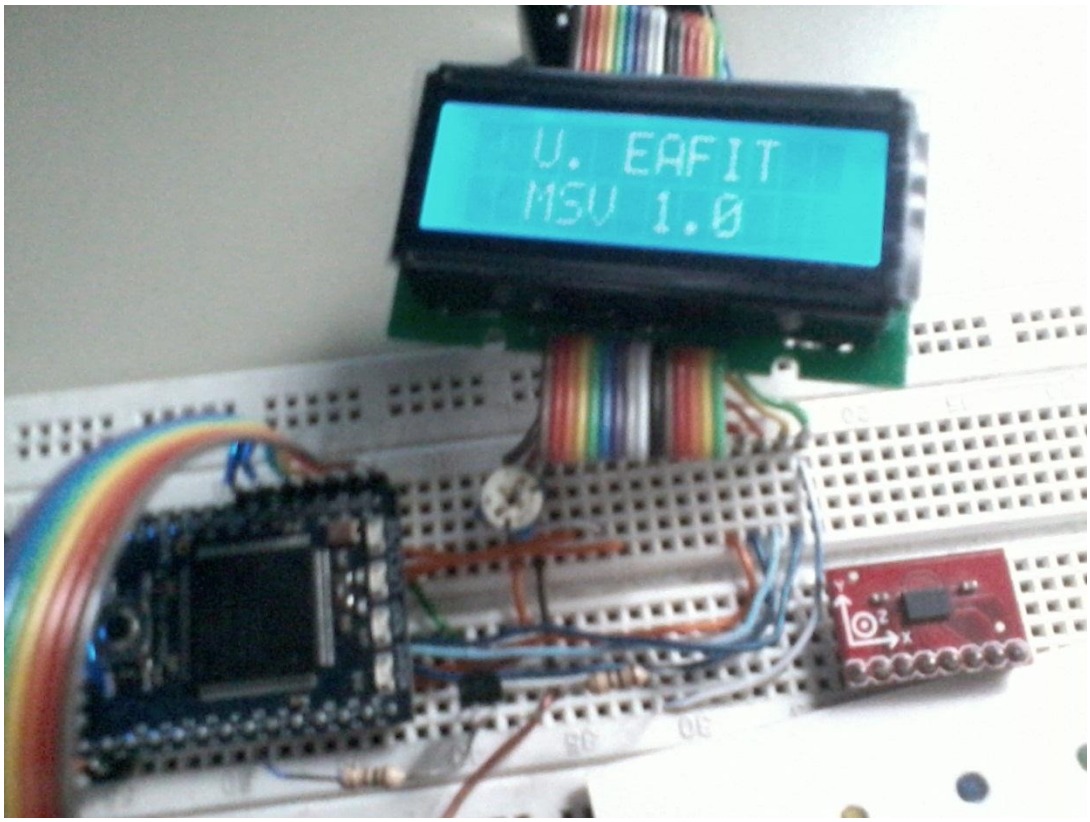
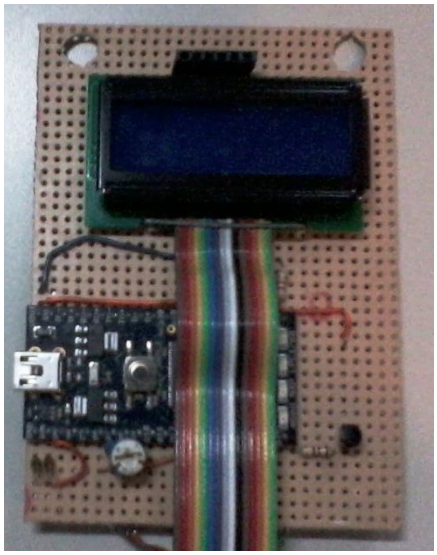


Figura 45. Montaje board universal

Sistema central



Punto de medición

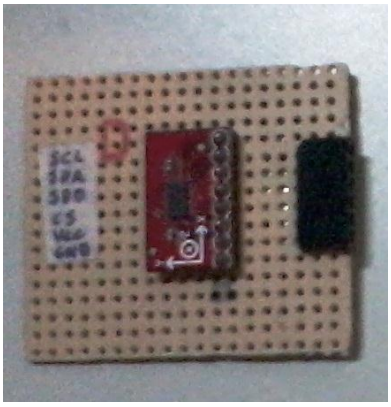


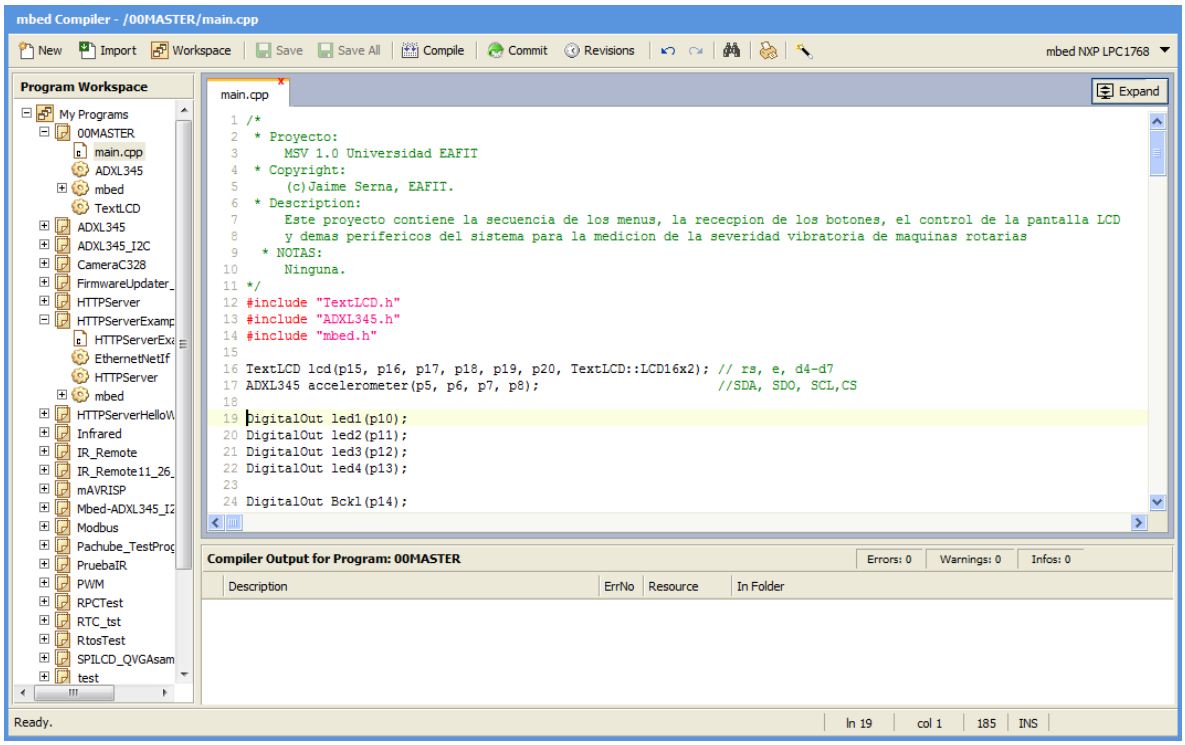
Figura 46. Prototipo final del sistema central

Luego de tener todo el hardware armado y funcionando, se realizan las pruebas de funcionamiento de todos los módulos, para estas pruebas, se utilizaron las mismas rutinas de código usadas en las pruebas de los circuitos virtuales, y se comprobó correctamente que todo el hardware del sistema estaba operativo y funcional, gracias a esto es posible continuar con la siguiente etapa del desarrollo del sistema.

7.6. PROGRAMACIÓN DEL SISTEMA

En esta etapa se realizó la programación de todas las rutinas de código del sistema, se programó el sistema de menús presentado en el ANEXO C – Concepción del sistema y además se hicieron todas las pruebas de funcionalidad posible, teniendo siempre en cuenta el documento de especificaciones y las funcionalidades que debía tener el equipo. En el ANEXO I – Programación del sistema al final de este documento, se encuentran los detalles de la programación y los códigos fuentes finales implementados en el sistema final.

Figura 47. Interfaz del software de programación del sistema



7.7. PRUEBAS, DEPURACIÓN Y FINALIZACIÓN

En esta etapa se le realizaron una serie de pruebas al sistema, entre estas están:

Pruebas en reposo.
Pruebas con vibraciones controladas.
Pruebas de medición en un banco de pruebas de desbalanceo.

En el Anexo J. Pruebas, depuración y finalización se encuentra la descripción de cada una de estas pruebas y los resultados obtenidos, a medida que se realizan las pruebas, se caracteriza el sistema y se hacen los documentos requeridos por el usuario, para este caso se generaron los documentos:

Hoja técnica del sistema
Informe de desarrollo del sistema
Manual de usuario
Informe de pruebas
Documento con el procedimiento de chequeo del equipo

Como punto de finalización del sistema se toma una tabla que se encuentra dentro del informe del desarrollo del sistema, la cual contiene el costo total de producción de un equipo de estas características, a partir del cual es posible visualizar la gran diferencia de precios con un equipo comercial de mantenimiento predictivo que realiza las mismas funciones.

Tabla 10. Costo producción Punto de medición

Ítem	**Descripción**	**Costo**
Acelerómetro	Sensor que digitaliza la señal de vibración	$ 80.000
Fabricación y ensamble de la tarjeta	Producción y soladura de la tarjeta del punto de medición	$ 50.000
Carcasa	Protección metálica punto de medición	$ 60.000
	Total Punto de medición	**$ 240.000**

Elaboración propia

Tabla 11. Costo producción Sistema central

Ítem	**Descripción**	**Costo**
Mbed	Controlador principal del sistema central	$180.000
Componentes varios	Periféricos y conectores de la tarjeta: Cristal, resistencias, capacitores, etc.	$ 50.000
Fabricación y ensamble de la tarjeta	Producción y soladura de la tarjeta del sistema central	$ 70.000
Carcasa	Protección sistema central	$ 120.000
	Total Sistema central	**$ 410.000**

Elaboración propia

7.8. EVALUACIÓN.

Una vez finalizado el desarrollo del sistema, se hace la entrega oficial al grupo GEMI, quien realiza la evaluación del sistema con base en el documento de especificaciones, en el ANEXO K. Evaluación se muestran la tabla de evaluación del sistema y el acta de entrega del sistema.

Con esta etapa se finaliza el desarrollo de un sistema electrónico de bajo costo para el mantenimiento predictivo de las PYME, las conclusiones y resultados del desarrollo de este proyecto se muestran en el siguiente capítulo.

CONCLUSIONES

Durante la revisión bibliográfica se encontró que a pesar de existir textos sobre sistemas embebidos que ya no aplican completamente a los sistemas actuales, se mantiene las tendencias predichas hace varios años, lo cual indica que los esfuerzos de la humanidad tienden cada vez a sistemas con más capacidades, más baratos y más incrustados en los elementos de uso diario, generando así mayor facilidad y confort para el ser humano en su vida diaria.

El nivel de evolución de los sistemas electrónicos tiene una tasa de crecimiento exponencial lo cual se ve reflejado en la rápida obsolescencia que adquieren los textos relacionados con esta área, se observa que cada vez se busca desarrollar sistemas más complejos y más rápidos, evolucionando las metodologías de desarrollo, buscando cada vez mayores niveles de abstracción y menores tiempos de salida al mercado.

Dado la extensa amplitud de la definición de los sistemas embebidos, se hace casi que imposible generar una metodología de diseño que sirva para cualquier dispositivo de esta área, lo cual se ve demostrado, que después de tantos años que existe esta rama, la mejor manera de desarrollar un producto es utilizar o ajustar un método existente según el tipo de sistema a desarrollar o de los recursos con los que se cuente.

Debido a lo volátiles que son los mercados de consumo y equipos electrónicos, se busca cada vez métodos y protocolos de diseño en los que el tiempo de desarrollo sea mucho menor, de modo que se pueda aprovechar un mayor tiempo de producto en el mercado, antes de que entre en estado de obsolescencia ante alguna nueva tecnología o producto que supere en costo y aplicación el producto

desarrollado, esto hace que el mercado electrónico sea de gran inversión, alta flexibilidad en el cambio y tiempos de vida de productos muy bajos.

El uso del protocolo para el desarrollo de un producto electrónico arrojo buenos resultados, además de poder llevar el seguimiento del estado de desarrollo de un sistema, se observa que este protocolo es útil para la implementación de sistemas que utilizan tecnología de bajo costo y cuya principal función es la medición de datos de un transductor, pero a su vez no es el mejor protocolo a seguir para los casos en que se trabajen dispositivos cuya principal función sea totalmente interacción con humanos y que no necesiten de mucho hardware.

Al generar un primer producto para el mantenimiento predictivo, se observa que la brecha tecnológica entre los productos comerciales que se deben importar para este fin, no es muy grande, por el contrario, este tipo de iniciativas generan que se aprovechen los avances tecnológicos y se puedan generar productos cada vez de mayor calidad y que estén en capacidad de competir con los productos exteriores, conservando siempre el bajo costo.

La industria Colombiana posee fuertes lazos costumbristas respecto a la tecnología en sus empresas, lo cual se observa con la tendencia de seguir utilizando las mismas tecnologías grandes y análogas, que no siempre son las mejores, lo cual genera una mayor brecha tecnológica con los países en desarrollo que cada vez más aprovechan las bondades que traen consigo los nuevos desarrollos electrónicos.

BIBLIOGRAFÍA

AHMED A. Jerraya, "Long Term Trends for Embedded System Design," Digital Systems Design, on Euromicro Symposium, pp.20-26, Euromicro Symposium on Digital System Design (DSD'04), 2004, ISBN: 0-7695-2203-3. [Articulo].

ANG, Simon. "Power-switching converters". Estados Unidos: Marcel Dekker Inc. 1995, ISBN: 0-8247-9630-6, 412p. [Libro].

ARNOLD, Ken, Embedded Controller Hardware Design, Estados Unidos: LLH Technology Publishing, 2000, ISBN 1-878707-87-6. 246p. [Libro].

ASHENDEN, Peter. Digital design an embedded systems approach using Verilog, Estados Unidos: Morgan Kaufmann, 2008, ISBN: 978-0-12-369527-7, 578p. [Libro].

BARR, Michael. Programing Embedded Systems in C and C++, Estados Unidos: O'Reilly 1Ed. 1999, 191p [Libro].

CABRERA, José, Ingeniería en Automática y Electrónica Industrial-Sistemas Analógicos, ULPGC. mayo de 2010, disponible en: http://www.ulpgc.es/hege/almacen/download/29/29861/filtros.pdf

CANTONE, DANTE, biblia del programador. implementación y debugging, Buenos Aires: Mpediciones. 1ed, 2008. 320p [Libro].

CEMB-HOFMANN, N500, [en línea] www.cemb-hofmann.com/pdf/N500_esp.pdf [citado 28 de abril de 2010]

CHIODO, M.; GIUSTO, P.; JURECSKA A.; HSIEH H.; SANGIONVANNI-VINCENTELLI A.; LAVAGNO L. Hardware/Software Co-design of Embedded Systems. En: IEEE Micro, 14(4): 1994. p. 26-36 [Articulo].

CORDEIRO, Lucas; MAR, Carlos; VALENTIN, Eduardo; CRUZ, Fabiano; PATRICK, Daniel; BARRETO, Raimundo; LUCENA, Vicente; A Platform-Based Software Design Methodology for Embedded Control Systems: An Agile Toolkit, 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, abril de 2008. ISBN: 978-0-7695-3141-0. [Articulo].

CTC, Tecnical info, Sensor Power Requirements, mayo de 2010, Disponible en: https://4-20ma.org/FileUp/3DNEWPDF/SensorPowerRequirements_InfoPage.pdf

DÍAZ, Patricia. Diseño de producto, Departamento de Sistemas y Computación del Instituto Tecnológico de La Paz, Julio de 2009, disponible en: http://sistemas.itlp.edu.mx/tutoriales/produccion1/tema3_1.htm

EMERSON PROCESS, CSI 2130 [en línea] *www.documentation.emersonprocess.com/.../2130_alldse_csi2130machineryhe.pdf* [citado 28 de abril de 2010]

ESCUDERO, J; PARADA M; SIMÓN F. Universidad de Sevilla, Plan Docente de la Asignatura: Instrumentación Electrónica, noviembre de 2006, disponible: http://www.dte.us.es/ing_inf/ins_elec/temario/Tema%204.%20Filtros.pdf

ETENGABEKO Ikaskuntza, La creación de los microprocesadores, articulo web [Internet]. Abril de 2010. Disponible en: http://www.hiru.com/es/teknologia_berriak/teknologia_01450.html. [Web]

Fundacion OPTI, Tendencias y aplicaciones de los Estudio de Prospectiva Sistemas Embebidos en España. Septiembre 2009, disponible en: http://www.opti.org/publicaciones/pdf/texto131.pdf [Articulo].

GAJSKI, Daniel D; ABDI, Samar; GERSTLAUER, Andreas; SCHIRNER, Gunar; Embedded System Design Modeling, Synthesis and Verification. Estados Unidos: Springer, 2009. ISBN 978-1-4419-0503-1, 366p. [Libro].

GANSSLE, Jack. Embedded Systems, World Class Designs. Estados Unidos: newness 2008 ISBN 9780750686259. 568p. [Libro].

_____. ICE Technology Unplugged. Embedded Systems Programming, :103-109. 1999. Disponible en: http://www.embedded.com/1999/9910/9910sr.htm [Articulo].

GANSSLE, Jack; BARR, Michael. Embedded Systems Dictionary, Estados Unidos: CMP Books, 2003. 291p [Libro].

GONZALES, Liliana; URREGO, Germán. Modelo de requisitos para sistemas embebidos, Revista Ingenierías Universidad de Medellín, volumen 7, No. 13, pp. 111-127 - ISSN 1692-3324 - julio-diciembre de 2008/164 p [Articulo].

HENZINGER, Tom, SIFAKIS, Joseph. "The embedded systems design challenge". Proceedings of the 14th International Symposium on Formal Methods (FM), Lecture Notes in Computer Science, Springer, August, 2006, pp. 1-15. [Articulo].

INESSMAN, INGENIERÍA ESPECIALIZADA EN SISTEMATIZACIÓN DE MANTENIMIENTO, diagnostico predictivo [en línea] http://www.inessman.com/ [citado 28 de abril de 2010]

ISO, INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. Technical product documentation: Mechanical vibration of machines with operating speeds from 10 to 200 rev/s – Basis for specifying evaluation standards. Switzerland: ISO, 1974, 9p (ISO 2372).

_____. Technical product documentation: Mechanical vibration – Evaluation of machine vibration by measurements in non-rotating parts – Part 3: Industrial machines with nominal power above 15kW an nominal speeds between 120 r/min and 15000 r/min when measured in situ. Switzerland: ISO, 1998, 12p (ISO 10816-3).

JERRAYA, Ahmed A., WOLF, Wayne. "Hardware/Software Interface Codesign for Embedded Systems," Computer, vol. 38, no. 2, pp. 63-69, Feb. 2005, ISSN:0018-9162. [Articulo].

KAMAL, Raj. Embedded systems: architecture, programming and design. \Estados Unidos: McGraw Hill. 2008. [Libro].

KODAK, Accelerating Product Time to Market, 2009, Disponible en: http://www.kodak.com/global/mul/business/solutions/files/accelerating_product_time.pdf

KUMAR, Vipin. Why Embedded Systems are patentable?, Abril de 2010, disponible en: http://patentadvise.blogspot.com/2010/04/why-embedded-systems-are-patentable.html

LEON Gerardo, La ingeniería de software y los sistemas embebidos. *[Internet]. Abril 2010. Disponible en:* http://profesores.elo.utfsm.cl/~agv/elo326/1s06/lectures/La_ingenieria_de_software_y_los_sistemas_embebidos.ppt

MARWEDEL, Peter. Embedded Systems Design. Estados Unidos: Springer. 2006, 1 Ed. ISBN: 978-1402076909. 250 p [Libro].

MASSACHUSETTS INSTITUTE OF TECHNOLOGY. http://web.mit.edu/

MICROCHIP, Analog-to-Digital Converter Design Guide, mayo de 2010. Disponible en: http://ww1.microchip.com/downloads/en/DeviceDoc/21841c.pdf

_____, FilterLab Filter Design Software, Disponible en: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010007

MICROCHIP. PIC18F4523, Datasheet

MISIÓNPYME, Pymes en cifras, 2008 http://www.misionpyme.com/cms/content/view/107/41/

MUÑOZ, Rafael. Los planes de negocios internacionales, una realidad en Colombia, revista digital el poder del marketing, año 2009, disponible en: http://www.elpoderdelmarketing.com/edicion04/art-07-planes-negocios-colombia-realidad.html

NATIONAL SEMICONDUCTOR, LP2951, Datasheet

NOERGAARD, Tammy. Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers (Embedded Technology). Estadps unidos: Newnes, 2005. ISBN: 978-0750677929, 656 p. [Libro].

NUNES, C.J.; Jr. SILVA, D.C.D.; Jr. FERNANDES, A.O. Hardware-Software Codesign of Embedded Systems, en Integrated Circuit Design, 1998. Proceedings. XI Brazilian Symposium, ISBN: 0-8186-8704-5, Agosto 2002. [Articulo].

PALACHERLA, Amar; MICROCHIP TECHNOLOGY INC. Aplication Note 542: Implementation of Fast Fourier Transforms, 1997, 22p. Disponible en: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en011084

PRUFTECHNIK , Vibxpert [en línea]
http://www.pruftechnik.com/fileadmin/user_upload/COM/Condition_Monitoring/Products/VIBXPERT/Brochure/VIBXPERT_brochure_es.pdf [citado 28 de abril de 2010]

RODRÍGUEZ, Astrid. La realidad de la pyme colombiana desafío para el desarrollo, 2003
Disponible en:
http://www.fundes.org/LibreriaPublicaciones/Realidad%20Pyme%20Colombiana.pdf

ROMBERG, Justin; JOHNSON, DON. Aliasing. The connexions project. disponible en: http://cnx.org/content/m12973/1.3

SAAVEDRA, Pedro. Bases del mantenimiento predictivo y del diagnóstico de fallas en máquinas rotatorias. Universidad de Concepción, 2000

SANGIOVANNI-VINCENTELLI, Alberto; PINTO, Alessandro. An Overview of Embedded System Design Education at Berkeley. ACM Transactions on Embedded Computing Systems, Vol. 4, No. 3, August 2005 [Articulo].

_____._____. Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design, Proceedings of the IEEE, Vol. 95,No. 3, March 2007 [Articulo].

SEMAPI ARGENTINA S.A. DSP Logger MX 300, [en línea] http://www.dsplogger.com/ [citado 28 de abril de 2010]

ST MICROELECTRONICS, MC34063A Datasheet.

SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO, Guía de patentes Abril de 2008, disponible en: http://www.sic.gov.co/propiedad/Instructivo/Guias_PI/Guia_Patentes.pdf

SYNOPSYS, Virtual Prototyping , 2010 Disponible en: http://www.synopsys.com/Tools/SLD/VirtualPrototyping/Pages/default.aspx

ÚBEDA*, Benito.* Apuntes de: Sistemas embebidos (2009), Tema 1. Universidad de Murcia,[Internet],Abril de 2010, disponible en:http://ocw.um.es/ingenierias/sistemas-embebidos/material-de-clase-1/ssee-t01.pdf

UNIVERSIDAD DE VALENCIA, sistemas electrónicos de alimentación, tema 3: análisis de convertidores buck, boost y buck-boost, Mayo de 2010, Disponible en: http://www.uv.es/~esanchis/sea/pdf/tema3-SEA-03.PDF.

UNIVERSITY OF CALIFORNIA, BERKELEY. http://berkeley.edu/

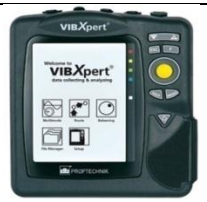
VALACHI, Joseph; GEORGE, Joey; HOFFER Jeffrey. Essentials of systems analysis and design, Estados Unidos: Pearson, Prentice hall, 3 Ed, 2006 [Libro].

NETRINO, Embedded Systems Glossary, http://www.netrino.com/Embedded-Systems/Glossary

ANEXO A. COMPARACIÓN Y COSTO DE EQUIPOS DE MEDICIÓN DE VIBRACIÓN

Equipo	Fotografía	Funciones	Característica técnicas	Rangos de medición	Precio
N500 [CEMB-HOFMANN]		- Medida de la vibración total (aceleración, velocidad, desplazamiento) - Medida de la fase de la vibración - Análisis de la vibración en frecuencia - Control de la vibración total en función del tiempo, o de la velocidad (diagrama de Bode) - Balanceo de ejes, Forma de onda	• ADC 24 bits • IP 65 • 2 puertos USB	• 10Hz a 20KHz • FFT: 100 a 3200 líneas	USD $ 14,000

CSI 2130 [*EMERSON PROCESS*]		- Toma y análisis de vibraciones
		- Balanceo
		- Alineamiento
		- Análisis de Canal cruzado
		- Análisis de Transitorio
		- Equilibrado dinámico
		- Alineación láser de ejes
		- Monitoreo del motores
			• ADC 16bits
			• IP 65
				• 10 Hz a 80KHz
				• FFT: 100-12800 líneas
				• Ventanas:
				  ○ Hanning
				  ○ Uniform
					USD $ 45,000

VIBXpert [PRUFTECHNIK]		- Colector de datos de vibraciones,
		- Medición de parámetros de proceso
		- Datos de inspección visual
		- Diagnóstico
		- Base de datos de rodamientos
		- Balanceo en 1 o 2 planos
			• ADC 12bits
			• IP 65
			• Compact Flash intercambiable
				• 0.5 Hz a 40 kHz
				• FFT: 100 a 102400 Líneas
				• Ventanas:
				  ○ Rectangular
				  ○ Hanning
				  ○ Flattop
					USD $ 25,000

DSP Logger MX 300 [SEMAPI]		- Colector de datos de vibraciones,
		- Analizador de Fase,
		- Analizador de campo FFT,
		- Balanceador de máquinas,
		- Medidor de Variables de Procesos
		- Medidor de Vibraciones Mecánicas
			• ADC 16/24 bits
			• IP 65
			• Puerto USB
				• 25Hz a 20KHz.
				• FFT: 400 a 4000 líneas
				• Ventanas:
				  ○ Rectangular
				  ○ Hanning
				  ○ Flattop
					USD $ 40,000

Comparación de equipos de medición de vibración

ANEXO B. ENTREVISTA DE CAPTURA DE REQUISITOS PARA EL DESARROLLO DE UN SISTEMA EMBEBIDO

Generalidades y funcionalidad del sistema

1. Describa las funciones y la composición del producto a desarrollar.

2. ¿Cuáles son las funciones puntuales que debe realizar el sistema?, p. ej. Medir, almacenar, transmitir.

Disponibilidad de objetos

3. Según el tipo de sistema a desarrollar, indique cuales serían los estado del sistema, p. ej. Encendido, apagado, midiendo, transmitiendo, etc.

4. ¿Cuáles serían los eventos para que el sistema cambie entre un estado y otro de los mencionados en la pregunta anterior?

5. ¿El sistema tendrá algún tipo de contraseña o existe la necesidad de proteger algún tipo de dato o función?

Convocación y demostración

6. ¿Cuál serían el modo de interacción o comunicación del sistema con el usuario?, p. ej. Botones, comunicación serial, pantalla, etc.

Selección de alternativas

7. ¿Si aplica, cuáles serían los menús principales del sistema? p. ej. Medir una señal, envío correo, reproducción de música, etc.

8. A partir de las funciones o menús anteriores, ¿cuáles serán las funciones segundarias o submenús en el sistema? p. ej. Adquirir datos, aplicar una transformada, configuración de la hora del sistema, etc.

Interacción de agentes

9. ¿Se comunica el sistema a desarrollar con otros sistemas o computadores?, ¿Cómo lo hace? (Protocolo, interfaz, estándar, etc.)

Acción del agente

10. Observando el sistema como una caja negra ¿Qué señales o comunicaciones o elementos entran al sistema? p. ej. Botones, comunicación serial, sensores.

11. Observando el sistema como una caja negra ¿Qué señales o comunicaciones o elementos salen del sistema? p. ej. Gráficos en pantalla, impresora, comunicación serial.

12. ¿En caso de que existan, cuáles serían las transformaciones que sufren las señales o datos desde la entrada hasta la salida?

Transferencia/actualización

13. Una vez terminado este sistema, ¿se pretende realizar nuevas versiones o actualizaciones a partir de este prototipo?

Interrupción/restitución/conservación

14. ¿Qué características de detección de fallas debe tener el sistema?, p. ej. Fallas en el sistema de alimentación, problemas en la ejecución del software, etc.

15. ¿Qué características de prevención de fallas debe tener el sistema?, p. ej. Monitoreo de la señal de alimentación, etc.

Desempeño/cambio

16. ¿Qué características de desempeño debe tener el sistema respecto a las funciones que hace? P. ej. No debe demorarse más de 10 segundos en almacenar 1000 datos, debe tener una frecuencia de muestreo de al menos 100KHz, etc.

17. ¿Cuál debe ser la disponibilidad o periodos de operación del sistema? P.ej. el sistema trabaja 24/7 o debe ser optimizado para periodos de 4 horas en trabajo continuo, etc.

18. ¿Qué tipo de prueba debería de aplicarse al sistema para evaluar su funcionamiento?

Precio y costos

19. ¿Cuál es el costo económico máximo manejable para este equipo? Descripción o caracterización de los agentes

20. ¿Qué protecciones especiales de seguridad y composición debe tener el sistema para el medio en el que trabajara? P. ej. Protección anti vandalismo, sellado contra agua, etc.

Parámetros del sistema

21. ¿Cuáles serían las características u hoja técnica del sistema final, tales como rangos de medición, cumplimiento de normas, etc.?

22. En caso de que el sistema tenga algún tipo de sensor, ¿Cuáles serán sus características (Resolución, precisión y sensibilidad)?

23. ¿Cuáles serían las características eléctricas del sistema?, p. ej. Portabilidad, duración de baterías (si las tiene), voltajes de alimentación, longitudes de los cables, etc.

Aspectos ambientales

24. ¿Cuáles son las condiciones ambientales del sitio de trabajo en el que operará el equipo? P. ej. Humedad, temperatura, etc.

25. ¿Qué aspectos medioambientales deben tenerse en cuenta para el producto a desarrollar?

Forma del sistema

26. ¿Cuáles serían las características físicas (Incluir diagrama si es necesario) que debe tener el sistema? P. ej. No debe medir más de 20 cm, ni pesar más de 1Kg, forma plana, etc.

27. ¿Qué características ergonómicas debe tener el producto a diseñar?

Tiempo de desarrollo

28. ¿En cuánto tiempo espera usted que se pueda desarrollar ese prototipo?

Información

29. ¿Qué tipo de documentación necesita del equipo a desarrollar una vez se ha finalizado el proyecto?

Mantenimiento

30. ¿Qué consideraciones se debe tener respecto al mantenimiento del equipo? p.ej. que no sea costoso, que no requiera conocimiento técnico, etc.

Se debe llegar a un acuerdo respecto a los siguientes temas:

- Aspectos relacionados a el ciclo de vida que no están incluidos en este protocolo tales como mantenimiento, uso y desecho del sistema a desarrollar
- Compromisos posteriores a la entrega del prototipo final
- Documentación requerida del sistema
- Aspectos legales del sistema (protección de propiedad intelectual)
- Sobrecostos en el cambio de las especificaciones
- Planos eléctricos y de hardware en general
- Programas del sistema
- Lista de entregables al final del proyecto

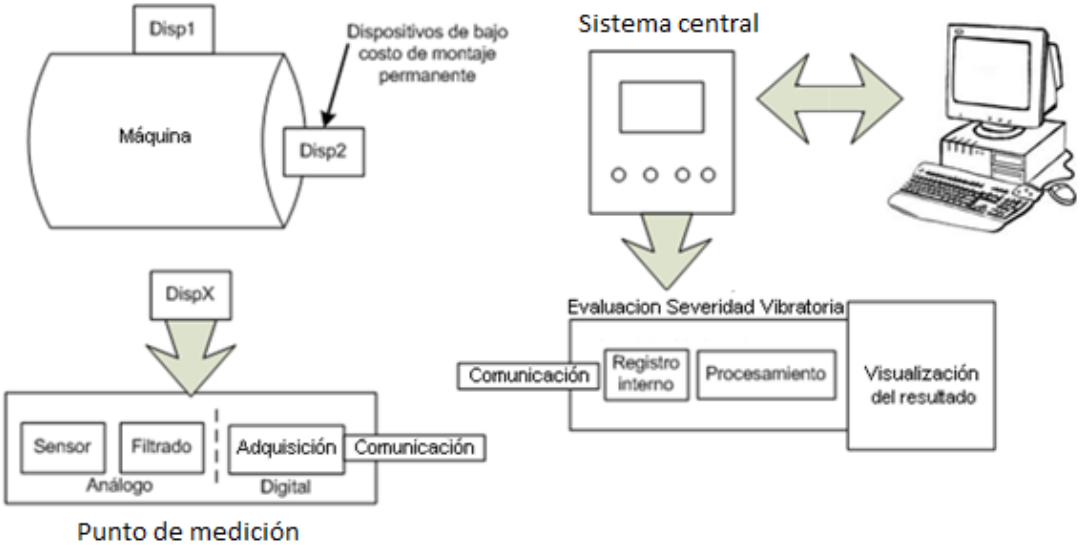
ANEXO C. CAPTURA DE LA ENTREVISTA

A continuación se muestra la información obtenida durante la entrevista realizada con un usuario del sistema a desarrollar.

Generalidades y funcionalidad del sistema

31. Describa las funciones y la composición del producto a desarrollar.

Se desea desarrollar un sistema que evalúe la severidad vibratoria por medio de las normas ISO 2372 e ISO 10816-3. Como componente nuevo, se requiere que el sistema esté basado en puntos de medición de bajo costo, los cuales puedan ser incrustados permanentemente en la máquina a medir, cada punto de medición contiene el sistema de acondicionamiento y digitalización de la señal, por otro lado, como sistema portable, se debe desarrollar un módulo central que se comunique con los puntos de medición, el cual realiza el procesamiento de los datos, almacenamiento y visualización de los resultados del sistema.



El dispositivo central debe contener un sistema de botones, una pantalla de indicación y cuatro luces de colores diferentes, a través de las cuales se indica en cuál de los estados de severidad vibratoria se encuentra la máquina, según la norma elegida. Se requiere que el sistema sea de fácil operación, de modo que una persona sin ningún conocimiento técnico pueda llevar a cabo la medición.

Una vez terminadas las Mediciones, se pueden descargar los resultados al PC a través de un cable USB, de modo que aparezcan, como si fuere una memoria, en un archivo de texto que pueda ser importado a una plantilla de Excel.

32. ¿Cuáles son las funciones puntuales que debe realizar el sistema?, p. ej. Medir, almacenar, transmitir.

- Filtrar
- Adquirir
- Comunicar
- Procesar
- Mostrar resultados
- Almacenar
- Disponibilidad en PC

Disponibilidad de objetos

33. Según el tipo de sistema a desarrollar, indique cuales serían los estado del sistema, p. ej. Encendido, apagado, midiendo, transmitiendo, etc.

- Desactivado
- Encendido

- Conectado
- Adquiriendo
- Procesando

34. ¿Cuáles serían los eventos para que el sistema cambie entre un estado y otro de los mencionados en la pregunta anterior?

    A través de la selección de opciones en pantalla por medio de los botones del usuario

35. ¿El sistema tendrá algún tipo de contraseña o existe la necesidad de proteger algún tipo de dato o función?

    No tendrá ninguna característica de este tipo.

Convocación y demostración

36. ¿Cuál serían el modo de interacción o comunicación del sistema con el usuario?, p. ej. Botones, comunicación serial, pantalla, etc.

    Cuatro botones para navegación y una pantalla alfanumérica para mostrar los menús, además, la configuración del sistema y la descarga de los resultados de medición, se hacen a través de la conexión con un PC.

Selección de alternativas

37. ¿Si aplica, cuales serían los menús principales del sistema? p. ej. Medir una señal, envío correo, reproducción de música, etc.

    MEDIR

CONFIGURAR

38. A partir de las funciones o menús anteriores, ¿cuáles serán las funciones segundarias o submenús en el sistema? p. ej. Adquirir datos, aplicar una transformada, configuración de la hora del sistema, etc.

MEDIR

Iniciar proceso de medición

CONFIGURAR

Seleccionar norma

Parámetros de medición

Interacción de agentes

39. ¿Se comunica el sistema a desarrollar con otros sistemas o computadores?, ¿Cómo lo hace? (Protocolo, interfaz, estándar, etc.)

La comunicación entre el sistema central y el punto de medición es alambrada, de igual forma el sistema central se comunica con el PC a través de cable USB.

Acción del agente

40. Observando el sistema como una caja negra ¿Qué señales, comunicaciones o elementos entran al sistema? p. ej. Botones, comunicación serial, sensores.

Al sistema entran señales de vibración a través de un acelerómetro e información del usuario a través de botones.

41. Observando el sistema como una caja negra ¿Qué señales o comunicaciones o elementos salen del sistema? p. ej. Gráficos en pantalla, impresora, comunicación serial.

    El sensor entrega una señal visual a través de una serie de LEDs de colores para mostrar los cuatro estados de la norma, además debe tener una pantalla alfanumérica para la visualización de los menús y un puerto de comunicación por USB para la conexión con el PC

42. ¿En caso de que existan, cuáles serían las transformaciones que sufren las señales o datos desde la entrada hasta la salida?

    Cálculo del valor RMS de la velocidad vibratoria para la evaluación de su severidad.
    Posiblemente conversión de la señal de aceleración a velocidad
    Posiblemente ejecutar la transformada de Fourier

Transferencia/actualización

43. Una vez terminado este sistema, ¿se pretende realizar nuevas versiones o actualizaciones a partir de este prototipo?

    Si, se desea aprovechar la modularidad del sistema.

Interrupción/restitución/conservación

44. ¿Qué características de detección de fallas debe tener el sistema?, p. ej. Fallas en el sistema de alimentación, problemas en la ejecución del software, etc.

    El sistema debe detectar cuando no hay comunicación entre los dispositivos

45. ¿Qué características de prevención de fallas debe tener el sistema?, p. ej. Monitoreo de la señal de alimentación, etc.

    Detectar cuando se va a hacer una medición y el acelerómetro no está conectado

Desempeño/cambio

46. ¿Qué características de desempeño debe tener el sistema respecto a las funciones que hace? P. ej. No debe demorarse más de 10 segundos en almacenar 1000 datos, debe tener una frecuencia de muestreo de al menos 100KHz, etc.

    Debe cumplir los tiempos normales de adquisición y procesamiento según los parámetros de adquisición seleccionados.

47. ¿Cuál debe ser la disponibilidad o periodos de operación del sistema? P.ej. el sistema trabaja 24/7 o debe ser optimizado para periodos de 4 horas en trabajo continuo, etc.

    El sistema debe durar mínimo 8 horas de trabajo continuo antes de que se descargue la batería por completo

48. ¿Qué tipo de prueba debería de aplicarse al sistema para evaluar su funcionamiento?

    Comparación del valor RMS de severidad vibratoria con la medición de otros equipos, además de realizar Mediciones correctas sobre un banco de pruebas de severidad vibratoria.

Precio y costos

49. ¿Cuál es el costo económico máximo manejable para este equipo? Descripción o caracterización de los agentes

    Al menos la mitad del costo del equipo comercial de más bajo costo, aproximadamente 2 millones de pesos

50. ¿Qué protecciones especiales de seguridad y composición debe tener el sistema para el medio en el que trabajara? P. ej. Protección anti vandalismo, sellado contra agua, etc.

    El sistema debe ser de una pieza, de fácil manipulación y fácil de limpiar

Parámetros del sistema

51. ¿Cuáles serían las características u hoja técnica del sistema final, tales como rangos de medición, cumplimiento de normas, etc.?

    Los indicados en las normas ISO de severidad vibratoria 10816-3 y 2372

52. En caso de que el sistema tenga algún tipo de sensor, ¿Cuáles serán sus características (Resolución, precisión y sensibilidad)?

    - Acelerómetros de bajo costo
    - Acelerómetros de 3 ejes
    - Con rango dinámico menor a ±50 g y mayor a ±10g

53. ¿Cuáles serían las características eléctricas del sistema?, p. ej. Portabilidad, duración de baterías (si las tiene), voltajes de alimentación, longitudes de los cables, etc.

    Cable de conexión del acelerómetro alrededor de 2 metros de longitud.

Aspectos ambientales

54. ¿Cuáles son las condiciones ambientales del sitio de trabajo en el que operará el equipo? P. ej. Humedad, temperatura, etc.

    Ambiente hostil industrial con presencia de motores y campos magnéticos

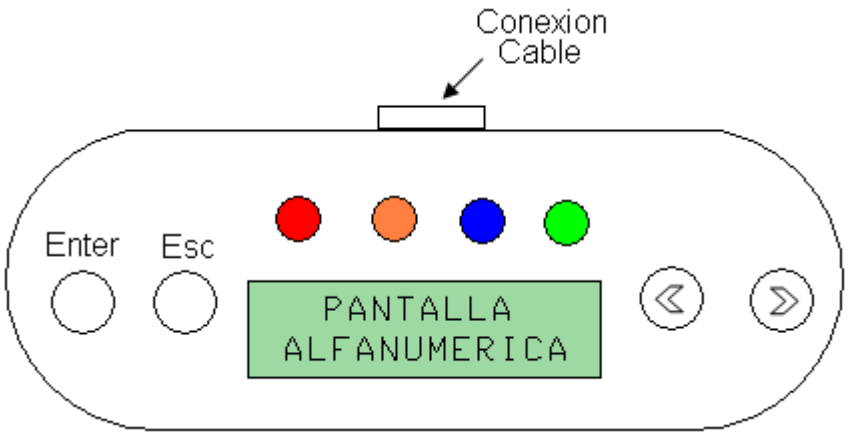
55. ¿Qué aspectos medioambientales deben tenerse en cuenta para el producto a desarrollar?

    Ninguno

Forma del sistema

56. ¿Cuáles serían las características físicas (Incluir diagrama si es necesario) que debe tener el sistema? P. ej. No debe medir más de 20 cm, ni pesar más de 1Kg, forma plana, etc.

    Sin ser estrictos el diseño físico del sistema puede ser del tipo:



El sistema debe tener un peso menor a 1 Kg

57. ¿Qué características ergonómicas debe tener el producto a diseñar?

El sistema debe ser de fácil manipulación para el usuario.

Tiempo de desarrollo

58. ¿En cuánto tiempo espera usted que se pueda desarrollar ese prototipo?

Alrededor de 6 meses

Información

59. ¿Qué tipo de documentación necesita del equipo a desarrollar una vez se ha finalizado el proyecto?

- Manual de operación.
- Documentación de desarrollo.
- Documento con el procedimiento del chequeo del equipo.

- Planos eléctricos y de carcasa del sistema.
- Ficha técnica.

Mantenimiento

60. ¿Qué consideraciones se debe tener respecto al mantenimiento del equipo? p.ej. que no sea costoso, que no requiera conocimiento técnico, etc.

    El sistema debe ser libre de mantenimiento.

Aspectos de común acuerdo.

Lista de entregables:

- Sistema completo totalmente operativo.
- Manual de operación.
- Documentación de desarrollo.
- Documento con el procedimiento del chequeo del equipo.
- Planos eléctricos y de carcasa del sistema.
- Programa ejecutable del PC.
- Códigos fuentes de todos los programas.
- Ficha técnica.

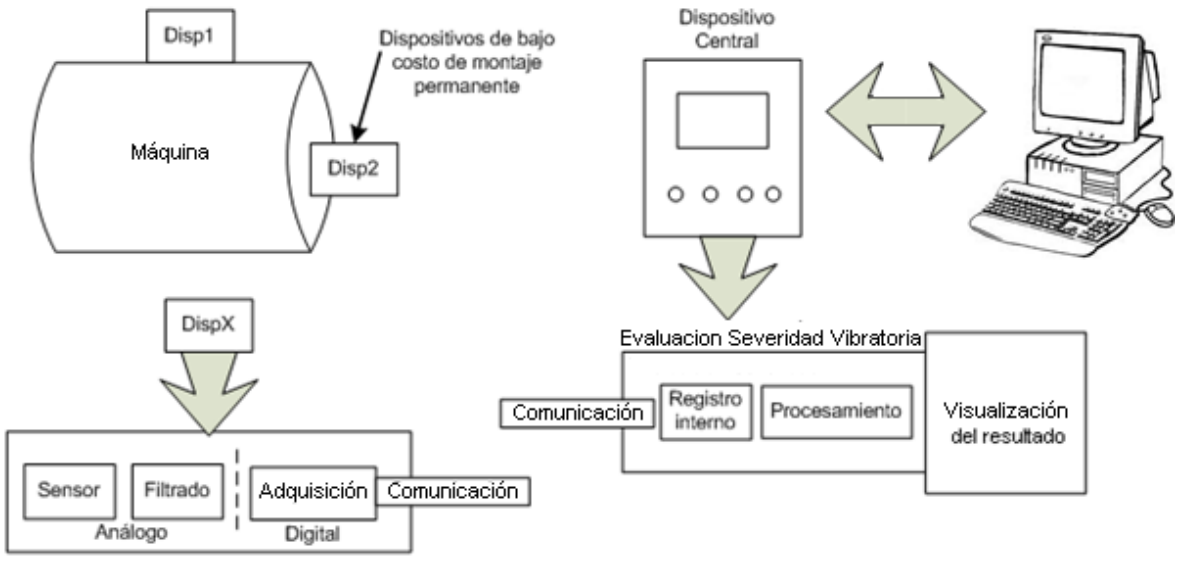
ANEXO D. DOCUMENTO ETAPA ESPECIFICACIONES

En este documento están plasmados los requisitos obtenidos del usuario para la realización del sistema embebido para la medición de la severidad vibratoria en máquinas rotativas.

NOMBRE DEL PRODUCTO: MSV (Medidor de Severidad Vibratoria)

DESCRIPCIÓN DEL SISTEMA:

Se desea desarrollar un sistema que evalúe la severidad vibratoria por medio de las normas ISO 2372 e ISO 10816-3. Como componente nuevo, se requiere que el sistema esté basado en puntos de medición de bajo costo, los cuales puedan ser incrustados permanentemente en la máquina a medir, cada punto de medición contiene el sistema de acondicionamiento y digitalización de la señal, por otro lado, como sistema portable, se debe desarrollar un módulo central que se comunique con los puntos de medición, el cual realiza el procesamiento de los datos, almacenamiento y visualización de los resultados del sistema.



El dispositivo central debe contener un sistema de botones, una pantalla de indicación y cuatro luces de colores diferentes, a través de las cuales se indica en cuál de los estados de severidad vibratoria se encuentra la máquina, según la norma elegida. Se requiere que el sistema sea de fácil operación, de modo que una persona sin ningún conocimiento técnico pueda llevar a cabo la medición.

Una vez terminadas las Mediciones, se pueden descargar los resultados al PC a través de un cable USB, de modo que aparezcan, como si fuere una memoria, en un archivo de texto que pueda ser importado a una plantilla de Excel.

CARACTERÍSTICAS DEL SISTEMA EMBEBIDO:

La siguiente es una lista de las características que debe cumplir el sistema embebido, bajo esta lista se debe realizar la evaluación del producto indicando si cumple o no cumple cada ítem.

1. Funcionalidad

    1.1. El sistema debe tener capacidad de adquirir señales de vibración

    1.2. El sistema debe calcular la severidad vibratoria de una máquina y realizar una clasificación según normas ISO 10816-3 o ISO 2372 según la elección del usuario

    1.3. El sistema mostrara el resultado de la evaluación de la severidad a través de una de cuatro luces de diferente color que representan los estados de la severidad vibratoria según la norma elegida.

1.4. El sistema debe estar en capacidad de conectarse a un PC para transferir los resultados obtenidos

1.5. Cada punto de medición solo deberá funcionar cuando se conecta el sistema central, de modo que cuando no hay conexión con este, el dispositivo este totalmente desactivado

1.6. El sistema debe indicar cuando no existe comunicación con los dispositivos.

2. Composición

2.1. El sistema estará compuesto por 2 tipos de sistemas diferentes:

2.1.1. Subsistema llamado punto de medición, en el cual se encuentra el transductor de aceleración triaxial, el acondicionamiento y la adquisición de los datos de vibración

2.1.2. Subsistema llamado sistema central que es donde se realiza el procesamiento de los datos y la visualización de los resultados

2.2. El sistema central deberá tener un método de visualización compuesto por 4 luces de colores para mostrar el estado de la máquina y una pantalla alfanumérica para el ingreso de los datos.

2.3. El sistema central deberá contener cuatro botones a partir de los cuales se realizara la navegación de menús en pantalla y el control del sistema.

2.4. El sistema completo deberá estar compuesto por 3 puntos de medición y un sistema central que evalúa la severidad vibratoria

2.5. El sistema central deberá ser portátil, y deberá operar con baterías recargables.

2.6. Los puntos de medición deberán ser diseñados para ser montados permanentemente en la máquina a medir

2.7. El sistema deberá contar con una carcasa que lo proteja de las condiciones ambientales a las que estará sometido

2.8. El tipo de comunicación entre los puntos de medición y el sistema central, así como de este último con el PC, será cableada.

3. Parámetros de operación

3.1. El sistema deberá tener una resolución de adquisición de datos de mínimo 12 bits.

3.2. El sistema deberá estar en capacidad de realizar Mediciones en un rango entre 2 y 1000 Hz según como lo dicta la norma

3.3. El sistema deberá estar en capacidad de operar en temperatura ambientales ente 0 – 70°C

3.4. La batería del sistema deberá tener una duración de mínimo 8 Horas y el proceso de cargado ser el estándar de las baterías comerciales.

4. Parámetros de sensores

4.1. El sistema deberá contar con un Acelerómetro cuyas características son:

4.1.1. Acelerómetro triaxial
4.1.2. Acelerómetro de baja frecuencia
4.1.3. Acelerómetros de bajo costo
4.1.4. Con rango dinámico menor a ±50 g y mayor a ±10g

5. Características físicas

5.1. El sistema deberá tener una carcasa que lo proteja del medio ambiente,

5.2. El sistema debe ser portable.

5.3. El sistema debe tener baterías recargables

5.4. El sistema debe tener un peso inferior a 1 Kg

6. Aspectos misceláneos Relevantes

6.1. La documentación del sistema consistirá en:

6.1.1. Manual de operación

6.1.2. Documentación de desarrollo

6.1.3. Documento con el procedimiento del chequeo del equipo

6.1.4. Planos eléctricos y de carcasa del sistema

6.1.5. Ficha técnica

Compromisos:

El desarrollo de este sistema se da por terminado una vez es aceptada la evaluación final del prototipo, los aspectos tales como mantenimiento, actualizaciones y mejoras no están incluidas en los requisitos de este proyecto, por lo cual se generarían como proyectos independientes a este.

Los aspectos de protección legal del prototipo correrán por cuenta del usuario final.

La documentación entregada solo será la que figura en este documento.

Cualquier cambio en los requerimientos del proyecto a partir de la 4 etapa del desarrollo, tendrán un costo fijado a partir de su magnitud y cambio en ejecución planeada del desarrollo del prototipo.

Lista de entregables:

1. **Sistema completo totalmente operativo**: prototipo funcional que evalúa la severidad vibratoria.

2. **Manual de operación**: documento que contiene la descripción de la correcta utilización del equipo.

3. **Documentación de desarrollo**: documento que contiene el desarrollo de cada una de las etapas realizadas durante la ejecución del proyecto.

4. **Documento con el procedimiento del chequeo del equipo**: documento que contiene la descripción de los pasos para comprobar que el sistema está operando correctamente.

5. **Planos electrónicos y de carcasa del sistema**: Dibujos descriptivos de la composición electrónica del sistema y de la composición de sus carcasas.

6. **Programa ejecutable del PC**: Programa a partir del cual se lleva la planilla de configuración y se hace la recepción de los datos de medición del sistema

7. **Códigos fuentes de todos los program**as: Archivos de programación del sistema en el lenguaje que se haya programado.

8. **Ficha técnica**: hoja técnica que contiene todos los parámetros y rangos del sistema.

ACTA APROBACIÓN DEL DISEÑO DE PRODUCTO

Los abajo firmantes reconocen que han revisado el documento de especificación de diseño del producto *ESV*, a partir del cual se realizará el desarrollo del producto y de acuerdo con el enfoque que se presenta. Cualquier cambio a esta definición de los requisitos deberá ser coordinada y aprobada por los abajo firmante o sus representantes designados,

Firma: _____ Fecha: _____

Nombre: Leonel Francisco Castañeda

Título: Dr. Ingeniero Mecánico

Cargo: Investigador principal

Firma: _____ Fecha: _____

Nombre: Germán René Betancourt G

Título: Msc. Ingeniero Mecánico

Cargo: Asistente de investigación

Firma: _____ Fecha: _____

Nombre: Jaime Alberto Serna

Título: Ing. Electrónico

Cargo: Asistente de investigación

ANEXO E. DOCUMENTO ETAPA CONCEPCIÓN DEL SISTEMA

En este documento se encuentra el desarrollo conceptual realizado para el sistema embebido, se inicia analizando las necesidades del usuario a partir de las cuales se realiza el diseño del flujo de la información, una vez que se conocen los procesos que se deben seguir, se ejecuta el diseño de bloques del hardware y software a partir de los cuales va a ser posible procesar la información requerida.
Como el objetivo de este equipo es medir la severidad vibratoria según las normas ISO 10816-3 e ISO 2372, a continuación se da un breve resumen sobre el contenido de estas normas para más adelante poder aplicar de forma clara su contenido al diseño del sistema.

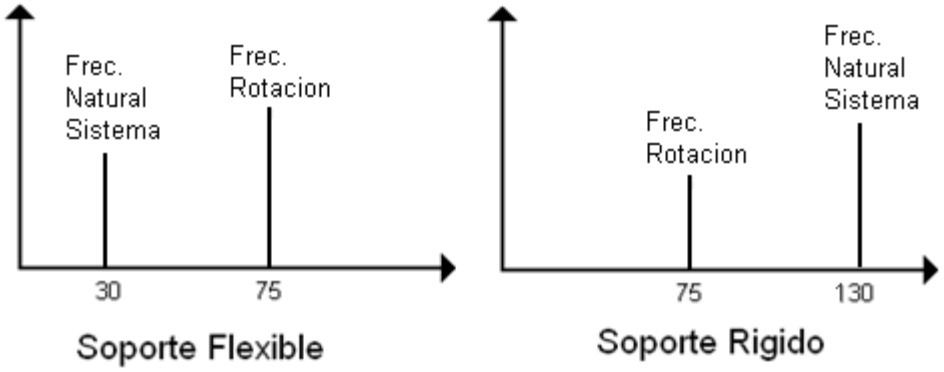
NORMA ISO2372: “Mechanical vibration of machines with operating speeds from 10 to 200 rev/s. Basis for specifying a evaluation standars”

Este estándar internacional define bases y reglas específicas a emplear en la evaluación de la vibración mecánica de máquinas que operan en el rango de frecuencias de 10 a 1000Hz. hay que tener en cuenta que las vibraciones medidas en la superficie de la máquina solo pueden proveer una indicación de los esfuerzos vibratorios dentro de la máquina, y no necesariamente reflejaran los esfuerzos reales de las partes críticas, ni tampoco aseguraran que no ocurran excesivos esfuerzos vibratorios, debidos por ejemplo. A resonancias locales. En particular, las vibraciones torsionales de las partes rotatorias no siempre generaran vibraciones medibles en la superficie de la máquina.

Los soportes de una máquina, se clasifican en dos tipos, soportes flexibles y soportes rígidos, en los soportes flexibles, la frecuencia fundamental del sistema

soporte-máquina es más baja que su frecuencia principal de excitación (en la mayoría de los casos es la frecuencia de rotación), para un soporte rígido, la frecuencia fundamental del sistema soporte-máquina es mayor que su frecuencia principal de excitación.

Tipos de soporte en una máquina.



Esta norma clasifica la máquinaria en 6 grupos diferentes dependiendo de la potencia de operación, el tipo de montaje utilizado y el uso de la máquina, esta clasificacion es mostrada en la siguiente tabla.

Tabla de clasificación de máquinas según la norma ISO 2372

CLASIFICACIÓN	POTENCIA
Clase I	<15kW
Clase II	15-75kW
Clase III	>75kW, Soporte rígido
Clase IV	>75kW, Soporte flexible
Clase V	Reciprocantes, desbalanceadas, Soporte rígido
Clase VI	Reciprocantes, desbalanceadas, Soporte flexible

El criterio de evaluacion de esta norma es el valor RMS de la velocidad vibratoria, donde a partir de una Medición de velocidad de la vibracion versus muestras en el tiempo, el valor RMS de la velocidad es calculado con la siguiente ecuacion.

$$V_{rms} = \sqrt{\frac{1}{T}\int_0^T V^2(t)\, dt}$$

Puesto que en las vibraciones, la aceleracion, la velocidad y/o el desplazamiento son determinados en funcion de la velocidad angular, es posible realizar trasformaciones de esta variable a traves de las siguientes relaciones.

$$V_{rms} = \sqrt{\left(\frac{1}{2}\right)\left[\left(\frac{\widehat{a_1}}{\omega_1}\right)^2 + \left(\frac{\widehat{a_2}}{\omega_2}\right)^2 + \cdots + \left(\frac{\widehat{a_n}}{\omega_n}\right)^2\right]}$$

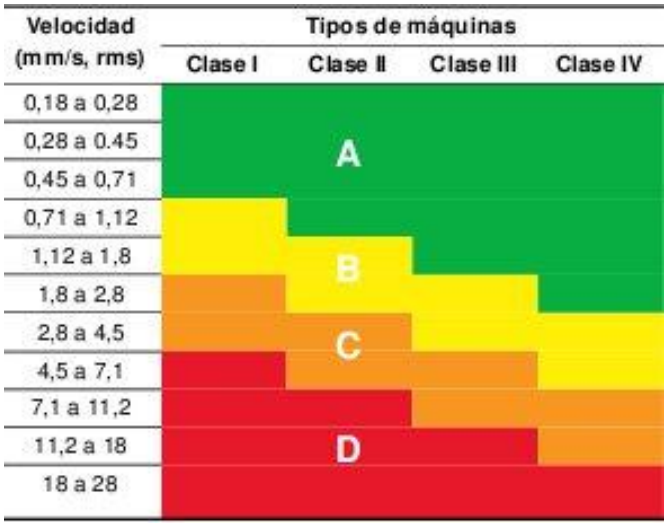
$$= \sqrt{\left(\frac{1}{2}\right)\left(\widehat{s_1}^2\omega_1{}^2 + \widehat{s_2}^2\omega_2{}^2 + \cdots + \widehat{s_n}^2\omega_n{}^2\right)}$$

$$= \sqrt{\left(\frac{1}{2}\right)\left(\widehat{v_1}^2 + \widehat{v_2}^2 + \cdots + \widehat{v_n}^2\right)}$$

De este modo, utilizando un acelerometro cuya salida esta dada en magnitudes de aceleracion es posible obtener el valor puntual del valor de velocidad RMS para ser comparado con la norma.

A partir del valor *RMS* de la velocidad de la vibracion y de la *Clase* de máquina, es posible clasificar la severidad vibratoria de la máquina con la siguiente tabla, las máquinas de las clases V y VI, son dificiles de calsificar debido a que ellas varian considerablemente sus caracteristicas vibratorias.

Tabla de clasificación de la severidad vibratoria según norma ISO 2372



SAAVEDRA, Pedro. Bases del mantenimiento predictivo y del diagnóstico de fallas en máquinas rotatorias, 2000.

Donde según la letra, la severidad vibratoria de la máquina es:

A: Buena

B: Satisfactoria

C: Insatisfactoria

D: Inaceptable

NORMA ISO10816: esta norma consiste en cinco partes bajo el titulo general: “Mechanical vibration. – Evaluation of machine vibration by measurement on non-rotating parts”.

- Part 1: General guideline
- Part 2: Large land-based steam turbine generator sets in excess of 50 MW
- Part 3: Industrial machines with nominal power above 15 kW and nominal speeds between 120 r/min and 15000 r/min when measured in situ
- Part 4: Gas turbine driven sets excluding aricraft derivates
- Part 5: Machine sets in hydraulic power generatin and pumping plants

- Part 6: Reciprocating machines with power rating above 100MW

Para este caso específico se utilizara la parte 3 de esta norma que aplican al conjunto de máquinas con potencias de más de 15KW y velocidad entre 120 rpm (2Hz) y 15000 rpm (250Hz), entre las cuales se presentan algunas turbinas a vapor, compresores rotatorios, bombas centrifugas, entre otras.

Al igual que en la anterior norma, en esta también es válido los tipos de soporte y también se presenta una clasificación en grupos, la cual es mostrada en la siguiente tabla.

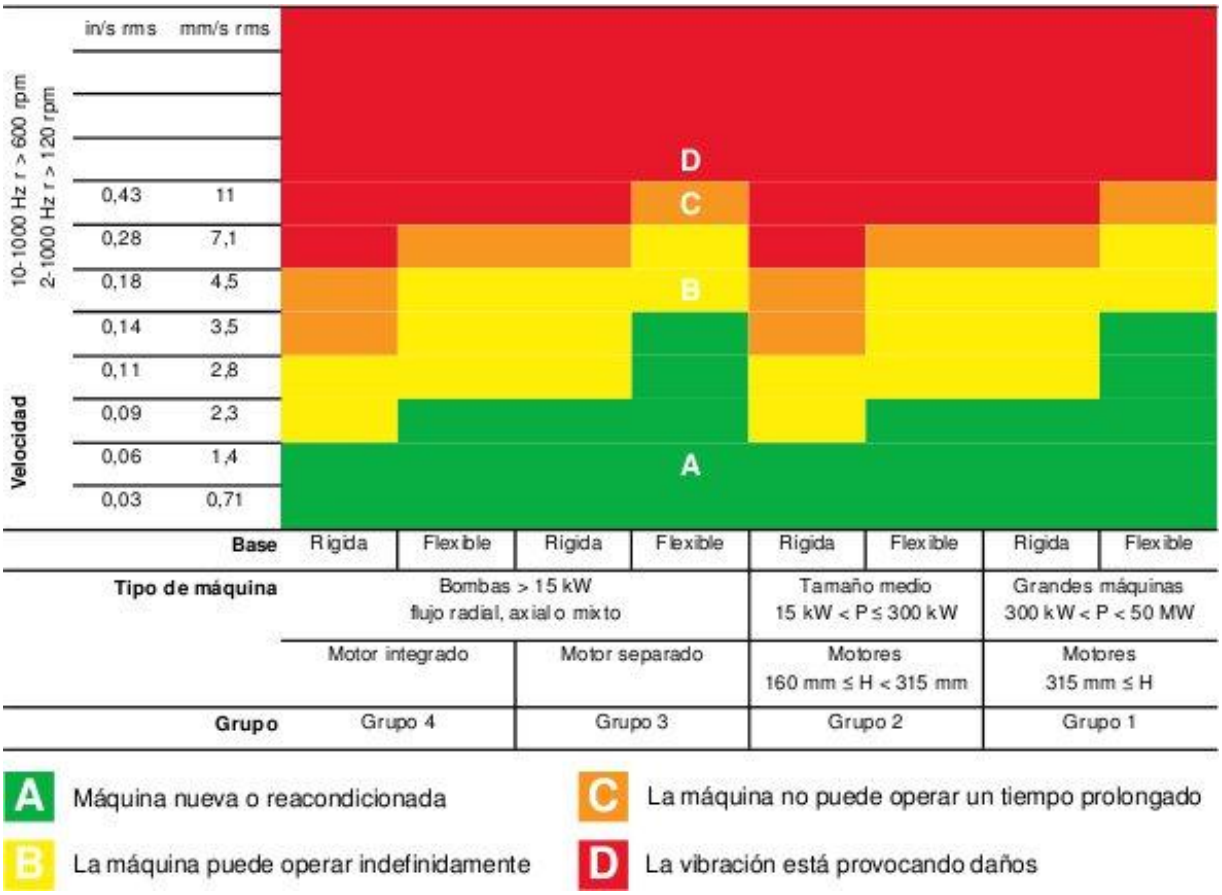
Tabla de clasificación de máquinas según la norma ISO 10816-3

CLASIFICACIÓN	POTENCIA
Clase I	300KW - 50MW
Clase II	15KW – 300KW
Clase III	Bomba >15Kw, motor separado
Clase IV	Bomba >15Kw, motor integrado

El criterio de evaluación de esta norma esta dado, al igual que la anterior, es el valor RMS de la velocidad de vibración, pero de igual forma también brinda los valores de comparación con los valores RMS de amplitud de desplazamiento de la vibración. Ambas magnitudes se relacionan a través de las ecuaciones mostradas en la norma anterior.

Esta norma también clasifica la severidad vibratoria de la máquina en 4 zonas diferentes, pero a diferencia de la anterior, proporciona una referencia un poco más descriptiva del estado de la máquina, a continuación se muestra la tabla de clasificación y luego la descripción de las zonas.

Tabla de clasificación de la severidad vibratoria según norma ISO 10816-3



SAAVEDRA, Pedro. Bases del mantenimiento predictivo y del diagnóstico de fallas en máquinas rotatorias, 2000.

Zona A: La vibración de máquinas nuevas o recientemente reacondicionadas puestas en servicio, normalmente debería estar en esta zona.

Zona B: Máquinas con vibración en esta zona son normalmente consideradas aceptables para operar sin restricción en un periodo de tiempo largo.

Zona C: Máquinas con vibración en esta zona son normalmente consideradas insatisfactorias para una operación continua para un tiempo prolongado, generalmente, estas máquinas pueden operar por un periodo limitado en esta

condición hasta que se presente una oportunidad conveniente para reparar la máquina.

Zona D: Los valores de la vibración de esta zona son considerados normalmente como suficientemente severos para causar daño a la máquina.

Cuando la máquina está en niveles de vibración mayores a los indicados en esta norma, se requiere que el fabricante de la máquina explique las razones de esto y en particular confirme que la máquina no sufrirá daño operando con valores vibratorios mayores.

La norma ISO 2372 es del año 1974, mientras que la norma ISO 10816-3 es del año 1998, si bien se considera una actualización de la norma anterior, ambas son ampliamente utilizadas, a continuación se muestra una tabla comparativa de las máquinas que cobijan cada norma y las clasificaciones que realiza cada uno a partir de la potencia y características de la máquina.

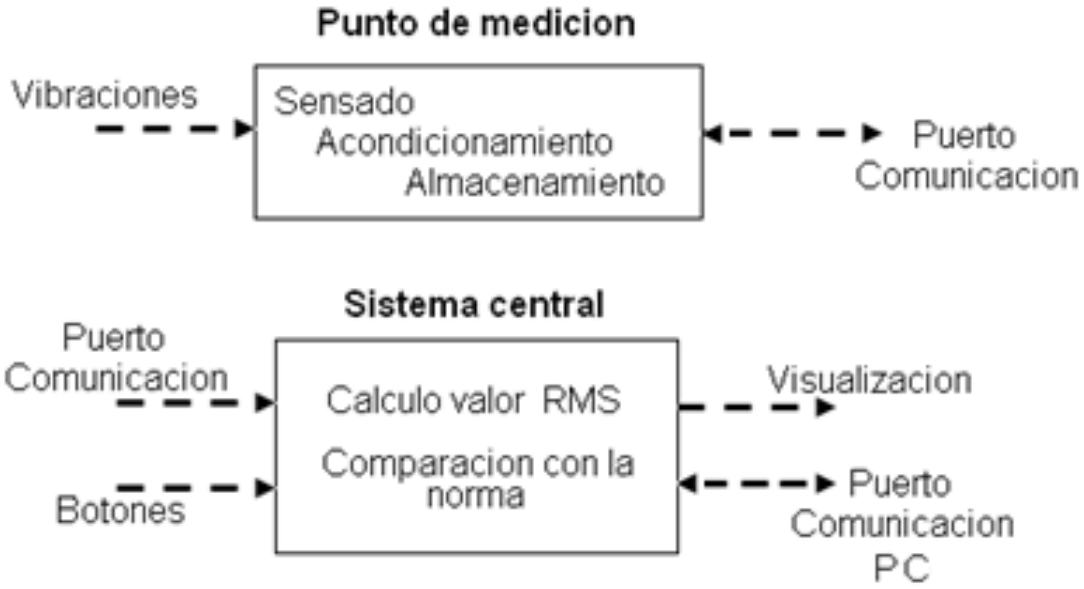
Tabla de comparación normas ISO 2372 e ISO 10816-3

Norma	ISO 2372	ISO 10816-3
Rango de revoluciones	60 rpm – 1200 rpm	120rpm – 15000rpm
Frecuencia de rotación	1 – 20 Hz	2 - 250Hz
Frecuencia instrumento	10 - 1000 Hz	10 - 1000 Hz
Clase I	<15kW	>300kW
Clase II	15-75kW	>15Kw-300kW
Clase III	>75kW, Sop. rígido	>15Kw, motor separado
Clase IV	>75kW, Sop. flexible	>15Kw, motor integrado

Diagrama de caja negra del sistema

A partir de las especificaciones del usuario, el diagrama de caja negra del sistema es el siguiente:

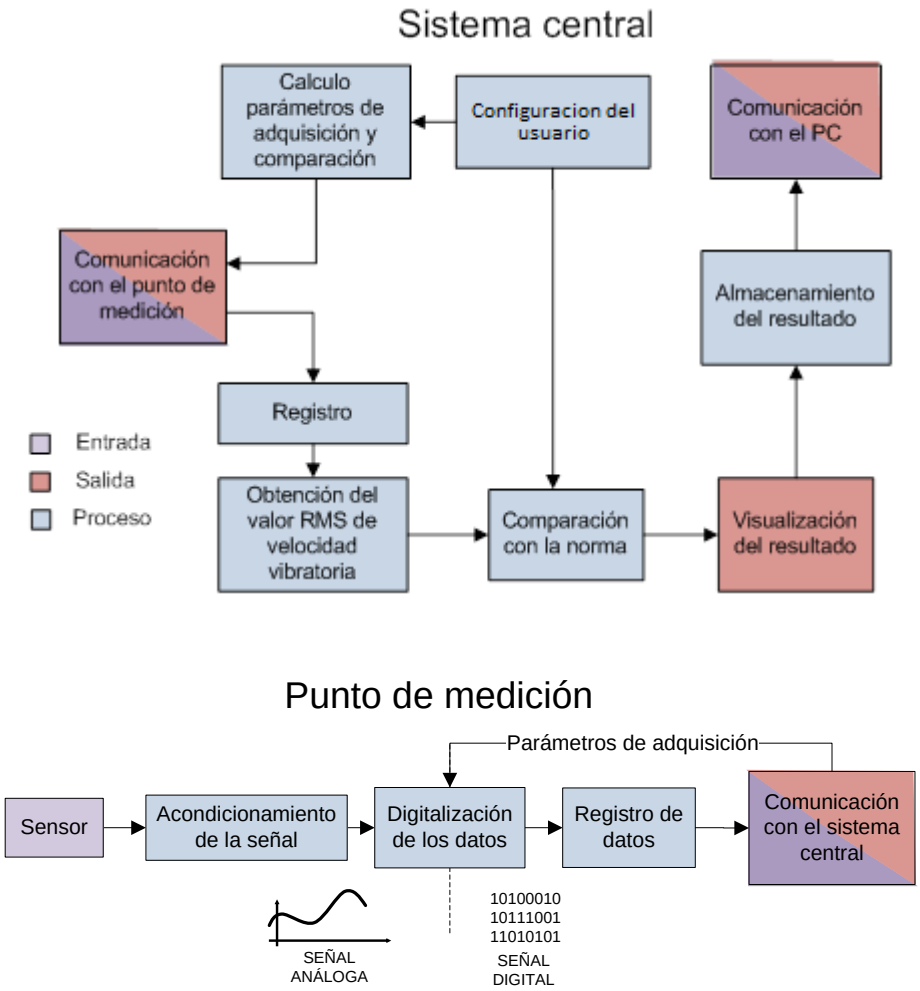
Diagrama de caja negra del sistema



Flujo de información del sistema

Una vez conocida la norma y todas las especificaciones por parte del usuario, referentes al sistema, se procede a realizar un diagrama de flujo de la información, en donde se busca mostrar de forma modular, cuales son los diferentes procesos o transformaciones que sufre la información en el sistema, a partir de las especificaciones, se tiene que el sistema está compuesto por dos módulos diferentes: un sistema llamado punto central donde se realiza la adquisición de datos y un sistema central donde se hace el procesamiento de la información según la norma, con base en esto, se generan un flujo para cada módulo, a continuación se muestra el diagrama:

Flujo de información en el sistema



Se observa que el sistema central depende de la configuración del usuario, la cual se guarda en el sistema, a partir de la cual se generan los diferentes parámetros de adquisición de datos y comparación con la norma para cada punto de medición, cuando el *punto de medición* es conectado al *sistema central,* se transfieren los parámetros de adquisición de datos con los cuales se realiza la digitalización de los datos.

Cuando el usuario desee comprobar la severidad vibratoria en una máquina, debe conectar el *punto de medición* al *sistema central* y luego seleccionar en éste

último, cuál de las dos normas quiere para llevar a cabo la evaluación, así como el número de máquina y demás parámetros para realizar la medición.

Luego de realizar la adquisición, El *punto de medición* almacena internamente los datos obtenidos, que son posteriormente transmitidos al *sistema central* el cual procede al cálculo del valor RMS de velocidad vibratoria, valor con el cual realizará la comparación con la norma utilizando los parámetros correspondientes al punto donde se realizó la medición, Una vez se hace la comparación con la norma, se procede a la visualización del resultado por medio de colores (tal como lo especificó el usuario) y posteriormente se realiza el almacenamiento de los datos requeridos por el usuario para cuando desee descargar la información en un PC.

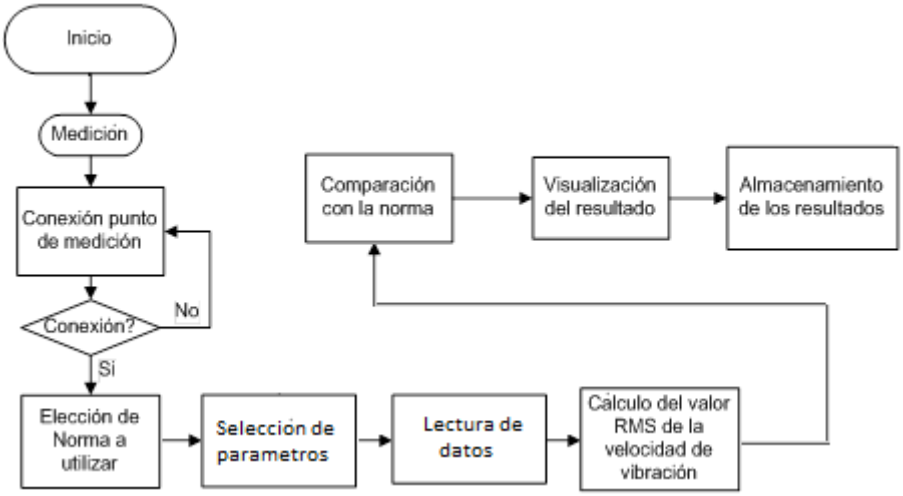
Del diagrama del flujo de la información, se puede observar que para el cumplimiento de las tareas del *Punto de medición* se necesita en su gran mayoría, módulos de hardware que cumplan esa funciones.

Para el *Sistema central* se observa que requiere un hardware de comunicación que de igual forma debe ser controlado por software, de igual forma para sus otras tareas debe ser necesario módulos de hardware controlados por software, por lo que cada módulo debe tener su desarrollo en cada aspecto, tanto hardware como software.

SOFTWARE

Del documento de requisitos consignado en el ANEXO C y del diagrama del flujo de información, se hace un diagrama que contiene las funciones principales de software a partir de las cuales será posible cumplir las funciones especificadas, se designan tres menús principales, los cuales requieren de una serie de funciones para llevar a cabo sus tareas, en el siguiente diagrama se muestra la forma general de las funciones necesarias y luego se prosigue con el desarrollo de cada una de ellas.

Diagrama general de software del sistema central



Funciones del sistema central

Medición

En esta función del sistema, el usuario podrá conocer la severidad vibratoria de una máquina bajo la norma ISO 2372 o ISO10816-3, para que esto sea posible son necesarias las siguientes funciones.

Conexión con el punto de medición

El sistema central debe garantizar que existe comunicación con el punto de medición, para esto, se deben generar una serie de pruebas donde se garantice que hay comunicación entre los puntos, pues de otro modo, no habría manera de garantizar que la medición es correcta y que si se tienen datos provenientes del acelerómetro.

Elección de la norma a utilizar

A través de la programación de esta función el usuario selecciona la norma con la cual desea hacer la evaluación de la severidad, la selección consistirá en mostrar

las dos normas en pantalla y resaltar la que usuario selección a través de los botones.

Selección de parámetros

Esta función genera los menús necesarios para que el usuario seleccione los parámetros de la máquina sobre la cual se va a realizar la medición, tales como clase de la máquina y tipo de soporte, una vez realizado esto, el sistema entra al estado de lectura de datos.

Lectura de datos

En este punto el sistema hace la lectura de los datos provenientes del acelerómetro y los almacena en memoria temporal para posterior procesamiento y evaluación de la severidad vibratoria del sistema.

Cálculo del valor RMS.

El proceso de aplicación de la norma inicia a partir de los datos de aceleración obtenido en la etapa anterior y el valor de frecuencia de muestreo de la tabla de parámetros de adquisición, debido a que la norma evalúa la severidad vibratoria a partir del valor de velocidad RMS, es necesario realizar esta conversión aplicación la formula.

$$V_{rms} = \sqrt{\left(\frac{1}{2}\right)\left[\left(\frac{\widehat{a_1}}{\omega_1}\right)^2 + \left(\frac{\widehat{a_2}}{\omega_2}\right)^2 + \cdots + \left(\frac{\widehat{a_n}}{\omega_n}\right)^2\right]}$$

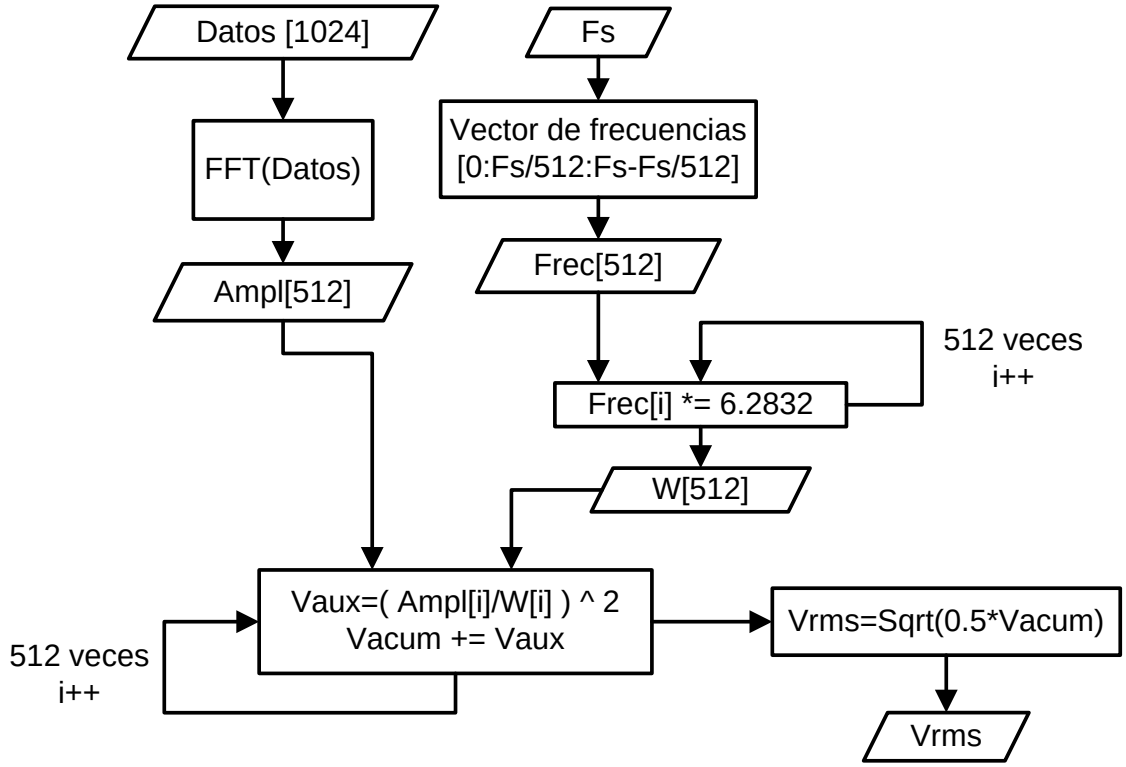
En donde cada componente a es la amplitud de aceleración y ω es la velocidad angular tomada del espectro de frecuencias de la señal adquirida, se observa que para poder obtener el valor de velocidad RMS es necesario calcular las componentes de frecuencia a través de la transformada de Fourier.

La transformada de Fourier genera un espectro con un número de líneas equivalente a la mitad de los datos ingresados, los equipos comerciales trabajan a partir de 100 líneas para realizar el análisis de la señal, para el caso de este equipo, se quiere trabajar con un total de 512 líneas del espectro de Fourier, por lo tanto, se hace necesario realizar la adquisición de 1024 datos de señal proveniente del acelerómetro, siendo esta cantidad de datos muy representativa para obtener una buena muestra de la señal de vibración.

El vector que se obtiene luego de realizar la transformada de Fourier a los datos de aceleración, contiene los valores de amplitud correspondientes a intervalos de frecuencia dentro del intervalo adquirido, el vector correspondiente a esas frecuencias se conforma desde 0 hasta Fs, con intervalos de Fs/512, este vector de frecuencias se multiplica por 2*π (6.2832) para obtener la frecuencia angular.

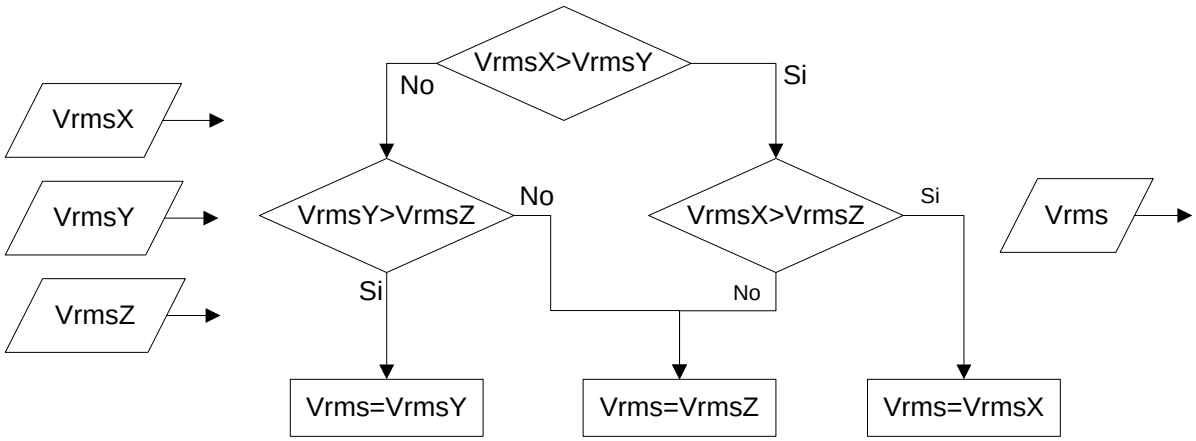
Una vez con estos dos vectores, se calcula primero el cuadrado de la división entre la amplitud y la velocidad angular y se lleva a una variable de acumulación, una vez todos los datos pasan este proceso, el acumulado total se multiplica por 0.5 y se saca su raíz cuadrada, obteniendo así, el valor RMS de la velocidad vibratoria, en la siguiente figura se ilustra el proceso para obtener este valor.

En la siguiente figura se muestra el algoritmo gráfico del proceso para obtener el valor RMS de la velocidad vibratoria partiendo de los datos recibidos del punto de medición y del valor de la frecuencia de muestreo con el que se obtuvieron los datos, se deja claro que este procedimiento se debe aplicar para los valores de cada uno de los 3 ejes del acelerómetro, obteniendo así los valores VRMSX, VRMSY, VRMSZ.



Comparación con la norma

La norma dice que para la evaluación de la severidad vibratoria en una máquina, se debe medir en varios puntos y direcciones, y tomar como dato característico de la máquina el valor RMS de severidad vibratoria más alto, para esto se utiliza un algoritmo simple de comparación, donde se obtiene el VRMS más alto:

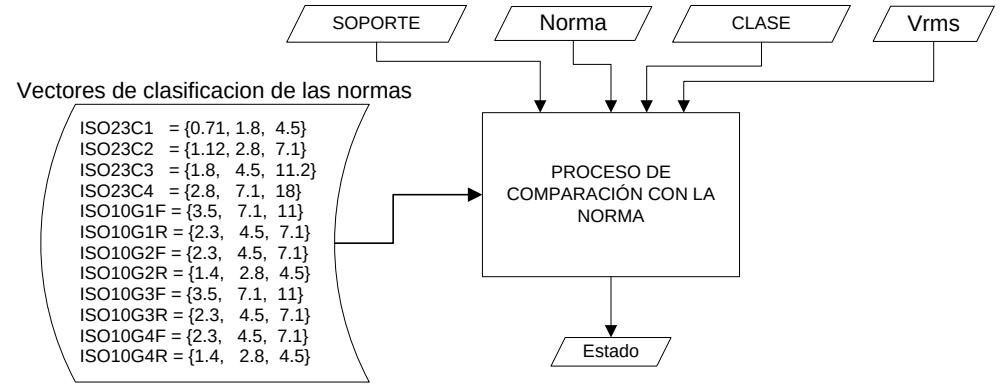


Una vez se tiene el valor RMS de severidad vibratoria característico de la máquina, se lleva a un proceso de comparación con una serie de vectores de datos construidos a partir las tablas presentada por cada norma, cada vector está compuesto por tres números que delimitan las cuatro zonas que presenta cada norma, los vectores correspondientes a la norma ISO 2372 inician con ISO 23, seguidos por la letra C de clase y un número entre 1 y 4 que indica a qué clase de máquina se refieren sus valores, para la norma ISO10816-3, inician con ISO10, seguido con la G de grupo, luego un número entre 1 y 4 que indica a qué clase de máquina se refieren sus valores y un F si el soporte es flexible o una R si el soporte es rígido, por ejemplo, el vector ISO10G2R corresponde a los límites de velocidad vibratoria de la norma ISO 10816-3 para una máquina clasificada en el grupo 2 con potencia entre 15KW y 300KW y que posee un soporte rígido. Sus valores son 1.4, 2.8 y 4.5, esto quiere decir que si el valor RMS de la velocidad vibratoria está por debajo de 1.4, la máquina tendrá una condición de *Buena*, si el valor obtenido se encuentra entre 1.4 y 2.8, estará en una condición *Satisfactoria,* si está entre 2.8 y 4.5 será *Insatisfactoria* y en caso que la velocidad vibratoria este por encima de 4.5, la máquina tendrá una condición de severidad vibratoria *inaceptable,* de este modo, a partir de estos vectores de comparación, es posible realizar la clasificación del estado de la severidad vibratoria de las máquinas incluidas en la norma.

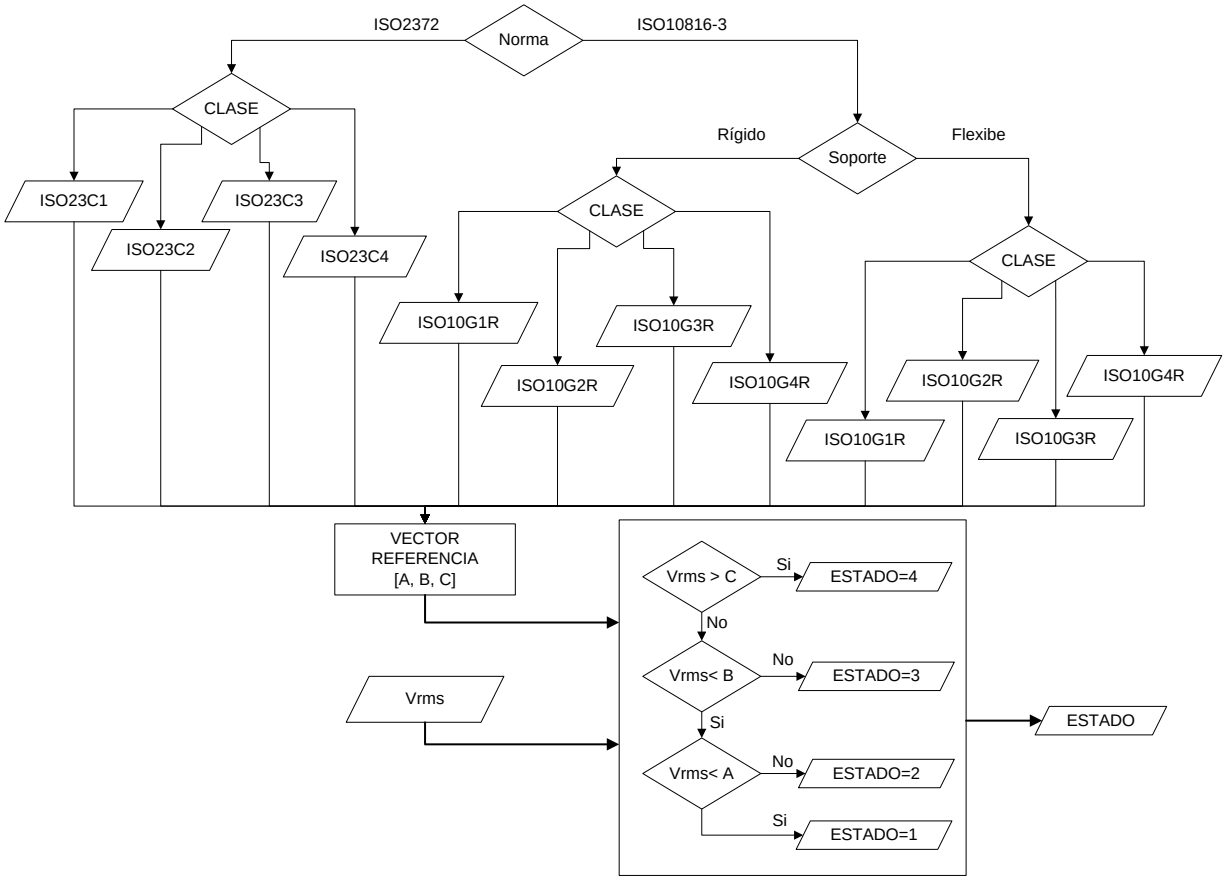
El proceso de comparación inicia a partir de la norma elegida por el usuario, de los datos del tipo de soporte y la clase de la máquina contenidos en la planilla de configuración del sistema, con estos parámetros se elige uno de los doce vectores para ser tomado como referencia para que el valor RMS de velocidad vibratoria sea comparado y se obtenga así, un resultado acorde a 4 valores diferentes de severidad vibratoria, en la siguiente figura se observa cómo se elige el vector de referencia que es ingresado a la siguiente etapa, en donde se compara el valor RMS de la severidad vibratoria, entre los 3 componentes del vector seleccionado,

y en base a esa comparación, se obtiene un resultado del estado de la severidad vibratoria en una máquina.

Aplicación de la norma



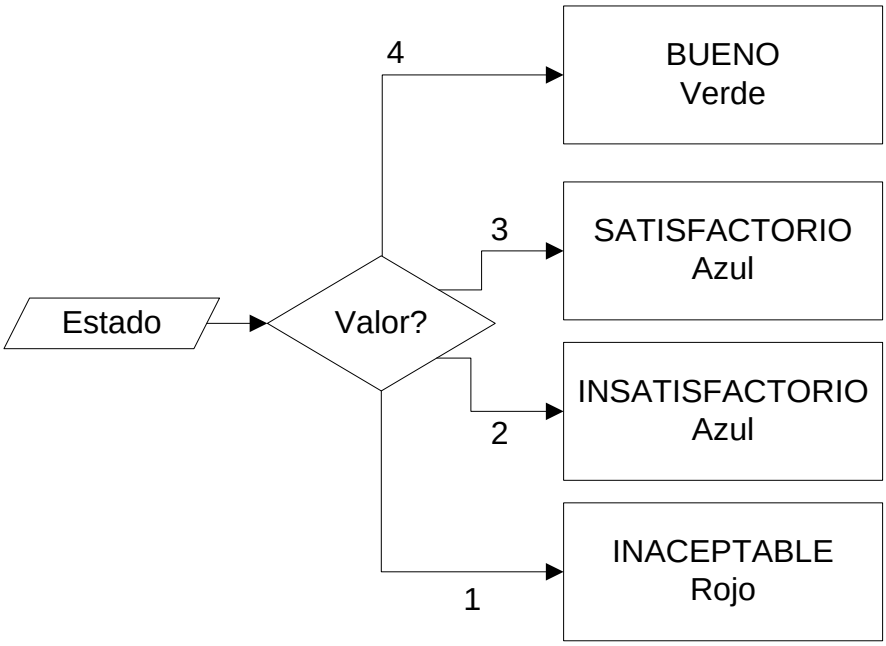
Proceso de comparación con la norma



Visualización del resultado

En esta etapa se muestra al usuario el resultado encontrado a partir de la evaluación de la severidad vibratoria, según la norma elegida, para esto se utilizara un sistema de indicación del estado por colores, en donde una luz del sistema indicara en cuál de los 4 estados se encuentra la máquina, adicionalmente, se mostrara en pantalla el veredicto de la norma, ya sea bueno, satisfactorio, insatisfactorio o inaceptable, el diagrama de este módulo seria:

Algoritmo de visualización del resultado

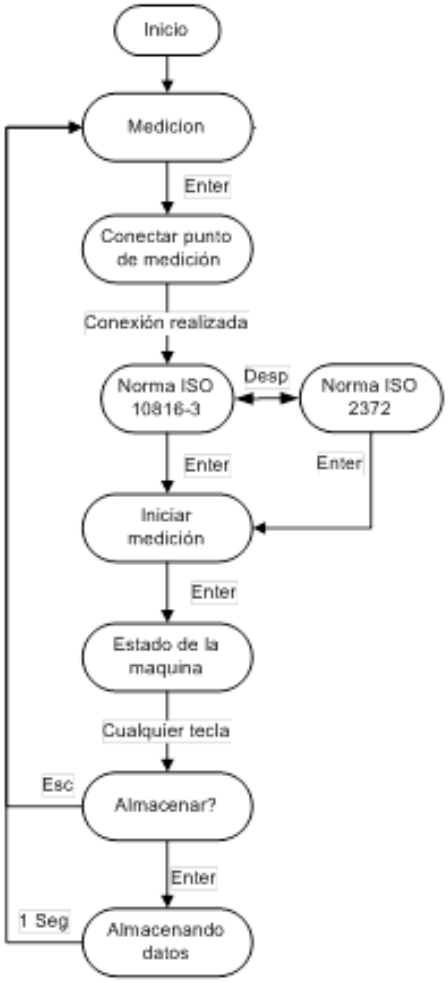


Almacenamiento de los resultados

Esta función debe organizar un vector de datos que será escrito en un archivo de texto separado por comas, de modo que a través de un macro en Excel pueda importar toda la información de las lecturas realizadas.

SISTEMA DE MENÚS

A continuación se muestra el diagrama simplificado de menús que tendrá el sistema.



HARDWARE

De la entrevista con el usuario del ANEXO C y del documento de requisitos consignado en el ANEXO D se genera una lista de requerimientos de hardware para poder cumplir con las funciones especificadas, entre estas tenemos:

El sistema estará compuesto por 2 tipos de sistemas diferentes:

- Subsistema llamado punto de medición, en el cual se encuentra el transductor de aceleración triaxial, el acondicionamiento y la adquisición de los datos de vibración

- Subsistema llamado sistema central que es donde se realiza el procesamiento de los datos y la visualización de los resultados

El sistema central deberá tener:

- Cuatro luces de colores para mostrar el estado de la máquina

- Una pantalla alfanumérica para el ingreso de los datos.

- Cuatro botones a partir de los cuales se realizara la navegación de menús en pantalla y el control del sistema.

- Alimentación con baterías recargables

El sistema deberá tener tres 3 puntos de medición donde cada uno deber tener:

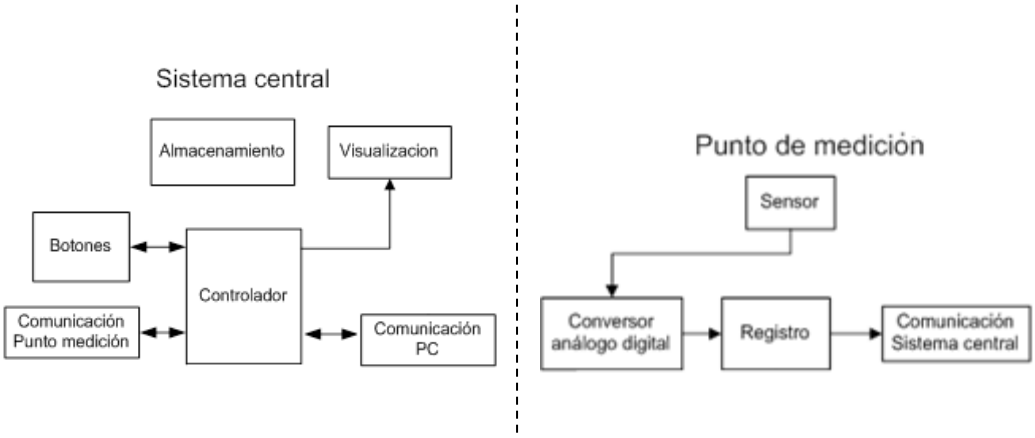
- El transductor de aceleración triaxial

- El acondicionamiento de esa señal

- Sistema de comunicación

A partir de estos requerimientos, de la descripción del sistema y del diagrama del flujo de información, se genera la siguiente gráfica que contiene los módulos de hardware para que el sistema cumpla sus tareas:

Diagrama de hardware del sistema



Este diagrama de hardware contiene todos los módulos necesarios para cumplir con las especificaciones realizadas por el usuario, para efectos prácticos, pueden existir dispositivos que contengan más de un módulo a la vez, de cualquier forma, se hace necesario especificar las características de cada uno para poder tener mejores parámetros de elección a la hora de asignar dichas tareas a módulos específicos.

Comunicación entre los dispositivos.

La comunicación entre el sistema central y el punto de medición, debe ser fiable, segura y robusta, se debe garantizar que los datos de la medición lleguen completos y en caso de detección de errores, se deben retransmitir los datos.

Puesto que se requiere que el punto de medición sea alimentado a través de la comunicación con el *sistema central*, se debe elegir un conector que permita tener dos líneas adicionales para suministrar el voltaje al *punto de medición.*

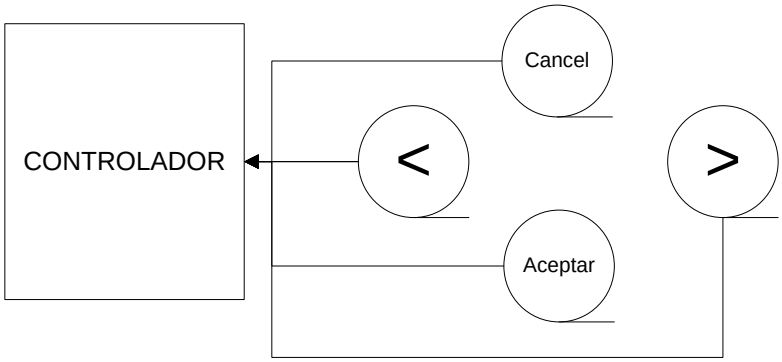
Módulos sistema central

Botones

Los botones le permitirán al usuario tener el control de la ejecución del sistema central, de modo que a través de estos, pueda indicarle la acción a realizar, ya sea el inicio de la medición, la importación de los datos de configuración o el envío de los datos almacenados al PC.

De las especificaciones del usuario, se utilizan 4 botones de los cuales 2 se utilizan para el desplazamiento de los menús, uno para aceptar el menú y otro para cancelar la acción,

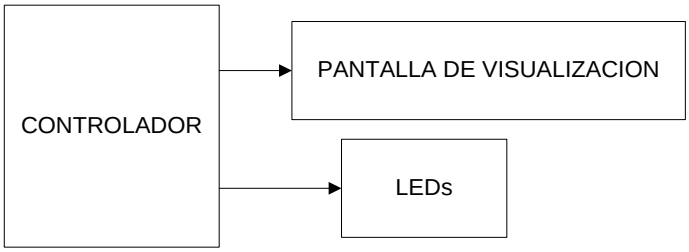
Diagrama de botones del sistema



Visualización

Este módulo del sistema debe estar compuesto por una pantalla de visualización que le permita al usuario interactuar con el sistema, de forma agradable y efectiva, además debe tener un pequeño sistema compuesto por 4 luces de indicación de diferentes colores que indiquen en cuál de los cuatro estados de clasificación propuestos por la norma, se encuentra la máquina que se está midiendo, el diagrama de esta etapa se basa simplemente en una pantalla y una serie de 4 LEDs controlados por diversos puertos del controlador principal.

Visualización del sistema



Comunicación PC

Del documento de especificaciones, se encuentra que el usuario desea tener comunicación con el dispositivo desde un PC a través de comunicación USB, para este tipo de comunicación, ya existe un estándar con varias opciones, por lo cual solo se debería elegir algunas características, tales como tipos de conectores, velocidad de transmisión, etc. De igual manera, existen una serie de dispositivos que generan las señales para la transmisión en este estándar, así como controladores que ya tienen inmersos ese tipo de dispositivos.

Almacenamiento

En este módulo del sistema se registran los datos de almacenamiento, debe ser rápido y confiable, puesto que los datos se necesitan solo durante el procesamiento, no tienen que ser guardados permanente, pero si debe tener una buena velocidad de almacenamiento de los datos.

Controlador

El controlador es el módulo principal del sistema, será el que coordine la operación de todos los módulos y realizara las operaciones sobre los datos, debe estar en capacidad de aplicar el algoritmo de la FFT a 1024 datos y tener la suficiente cantidad de pines para controlar los demás módulos, deberá tener también una entrada análoga con la cual podrá sensar el estado de la batería del sistema

Alimentación del sistema

El módulo de alimentación es el que suministra el voltaje a todo el sistema para que pueda operar correctamente, puesto que el sistema debe ser portable, este módulo debe estar compuesto por una serie de baterías que tengan la capacidad de suplir la carga que el sistema le genere por un tiempo mínimo de 8 horas en operación continua, debe tener un swiche mecánico de encendido y apagado a través del cual se aísle la batería del sistema, de modo que no tenga ningún consumo mientras el sistema no está en uso, este módulo debe tener un sistema apoyado en el controlador principal, que constantemente este evaluando el voltaje que entra al sistema y le informe al usuario cuando la batería esta próxima a descargarse completamente.

A partir del voltaje entregado por la batería se realiza una regulación de voltaje, de modo que se puedan establecer los distintos niveles de referencia que requiera el sistema, según las necesidades de alimentación de los diferentes componentes del sistema, en caso de que necesite más de uno.

Módulo punto de medición

Sensor

El sensor más apropiado para la medición de vibraciones, es un acelerómetro, a continuación se muestran algunas características tomadas del documento de requerimientos, a partir de las cuales se debe realizar la elección del sensor.

- Acelerómetro triaxial
- Acelerómetro de baja frecuencia
- Acelerómetros de bajo costo
- Con rango dinámico menor a ±50 g y mayor a ±10g

Esta etapa debe incluir el acondicionamiento de la señal proveniente del acelerómetro, la cual depende en gran parte de las indicaciones que entrega el

fabricante del sensor, según su principio de funcionamiento y condiciones de alimentación, por lo cual la configuración de esta etapa se hace una vez sea elegido el sensor

Conversor análogo digital

El módulo de conversión análoga digital, es el encargado de realizar la adquisición de datos con los parámetros configurados en el sistema para cada punto de medición, deberá tener una resolución de por lo menos 12 bits, y tener capacidad de entregar una buena velocidad de muestreo, la frecuencia máxima de adquisición que necesitaría el sistema, seria al medir una máquina que opere a 60.000 RPM, con lo que necesitaría tener una frecuencia de muestreo de 2.5Khz, si bien no es una frecuencia de adquisición muy alta, es un parámetro a tener en cuenta.

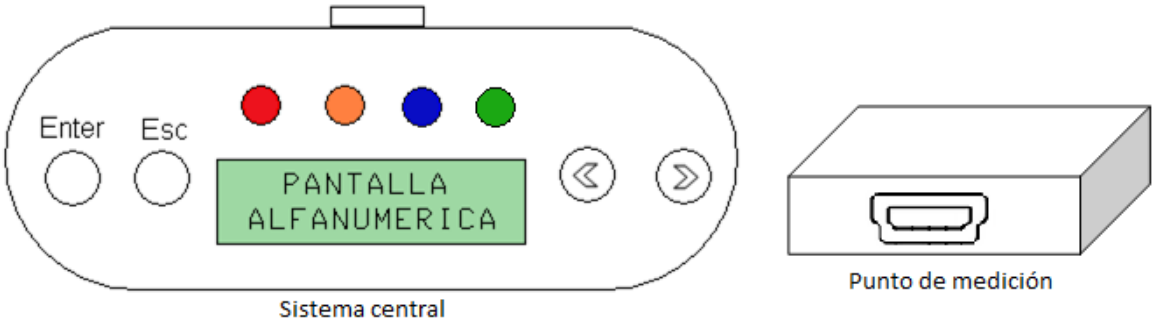
Registro

Este módulo tiene las mismas características que el módulo de almacenamiento del sistema central, sirve para registrar los datos de almacenamiento, debe ser rápido y confiable, puesto que los datos se necesitan solo mientras que se hace la transmisión, no tienen que ser de almacenamiento permanente, pero si debe tener una buena velocidad de almacenamiento, por la cantidad de datos, debe estar en capacidad de almacenar por lo menos 7Kbytes.

CARCASA DEL SISTEMA

En esta etapa del protocolo se busca generar la idea de la forma física de la carcasa del sistema, puesto que el usuario ya posee una idea esta y además posee una forma agradable y bien distribuida, no es necesario definir por ahora más aspectos relacionados con la carcasa, en la siguiente figura se muestra el diagrama inicial de la carcasa, la cual puede variar en algunos aspectos según las

necesidades y requerimientos para su fabricación una vez se levanten las características finales con base en el hardware construido de los dos sistemas.



CRONOGRAMA DE ACTIVIDADES

Teniendo en cuenta las definiciones anteriores se realizan una lista de actividades a seguir para el correcto avance del proyecto, se da un tiempo estimado de la duración de actividades considerando el tiempo promedio de desarrollo sin muchos inconvenientes, además de contar con la disposición de los componentes necesarios.

Actividad	Tiempo (semanas)												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Diseño Detallado													
Análisis de rendimiento													
Pruebas de módulos de hardware													
Pruebas de adquisición de sensores													
Implementación del hardware													
Programación del sistema													
Elaboración de carcasa													
Pruebas y depuración de sistema													
Evaluación del sistema													

ANEXO F. DOCUMENTO ETAPA DISEÑO DETALLADO

En esta etapa se busca diseñar al detalle todos los componentes de hardware y software del sistema en base al diseño bloques general del sistema realizado en la etapa anterior, a continuación se inicia el desarrollo del diseño del software para posteriormente realizar el del hardware.

SOFTWARE

A partir del diagrama de bloques diseñado en la etapa anterior, se plantean los algoritmos de las funciones necesarias para llevar a cabo la correcta ejecución del sistema, al final de este anexo se encuentra una lista de algoritmos en pseudocódigo a los cuales se les hará referencia durante esta etapa de diseño, cada algoritmo está debidamente numerado y titulado para evitar confusiones a la hora de hacer referencia.

Máquina de estados del sistema:

Como primer paso para el diseño detallado del software, se genera la máquina de estados del sistema, a partir de la cual se podrá llevar control sobre la ejecución del programa, dicha máquina consiste en una variable que contiene el estado del sistema y una serie de condicionales que controlan la ejecución de un conjunto de líneas de código, de ese modo, las tareas que ejecuta el sistema corresponderán únicamente al módulo que se esté ejecutando y los cambios de estado, se dan a partir de las elecciones del usuario y de la función que está ejecutando el sistema.

El Algoritmo 1 – Máquina de estados, en la lista de algoritmos al final de este documento, ilustra como es el manejo de la máquina de estados global del sistema, a partir de la cual se hacen los llamados a las funciones.

A partir de esta máquina de estados, se describen a continuación los algoritmos para los diferentes módulos de software del sistema tomando como referencia la gráfica *Diagrama general de las funciones del sistema central* del documento de concepción del sistema.

Funciones del sistema central

Conexión punto de medición:
Del diagrama de conexión del sistema central con el punto de medición realizado en la etapa anterior, se genera el algoritmo para realizar el intercambio de datos entre el sistema central y el punto de medición, dicho algoritmo reúne las características principales del sistema de comunicación que se quiere implementar, de igual forma la conformación de las tramas y demás aspectos reales a implementar, dependen específicamente del protocolo de comunicación que se elija, el pseudocódigo de este módulo de software se encuentra en el Algoritmo 2 – Conexión con punto de medición al final de este documento.

Elección de la norma a utilizar:
El Algoritmo 3 – Elección de la norma a utilizar muestra el código para seleccionar una norma cuyo número es mostrado en pantalla y se puede hacer un desplazamiento del menú, a través de los botones laterales, por medio de la tecla Enter, el sistema pasa al siguiente menú y almacena en una variable cual norma es la que se va a utilizar.

Enviar parámetros de adquisición:
Para esta función no se genera ningún algoritmo, pues se supone que el protocolo de comunicaciones establece un canal, esta función solo se encarga de enviar un dato entero que contiene el periodo de muestreo que debe tener el punto de medición.

Registrar los datos recibidos:

El algoritmo de este módulo consiste en almacenar en un espacio de memoria un vector de 6144 bytes los cuales son discriminados en los datos de tres ejes, donde cada uno tiene 2048 datos, en donde cada par de bytes conforman un dato de 16bits que contiene el valor de aceleración medida, este algoritmo es el Algoritmo 4 – Registro de datos de la lista de algoritmos al final de este anexo.

Cálculo del valor RMS de la velocidad vibratoria:

Para el cálculo del valor RMS se utiliza la ecuación:

$$V_{rms} = \sqrt{\left(\frac{1}{2}\right)\left[\left(\frac{\widehat{a_1}}{\omega_1}\right)^2 + \left(\frac{\widehat{a_2}}{\omega_2}\right)^2 + \cdots + \left(\frac{\widehat{a_n}}{\omega_n}\right)^2\right]}$$

En el Algoritmo 5 – Cálculo del valor RMS de velocidad vibratoria se muestra el procedimiento para obtener este valor, se hace la aclaración que para obtener las componentes ω_n de la medición es necesario realizar la transformada rápida de Fourier, pero debido a que existen muchas librerías que implementan esta función, se toma simplemente como una línea de código que genera dicho procedimiento

Aplicación de la norma:

En la lista de Algoritmos, el Algoritmo 6 – Aplicación de la norma, muestra todo el procedimiento en pseudocódigo de esta función, la cual está basada en el los diagramas mostrados en la etapa anterior de concepción del sistema, esta función utiliza los parámetros ingresados por el usuario y los datos digitalizados en la etapa anterior, a partir de los cuales, realiza una clasificación según la norma y entrega un valor de severidad vibratoria.

Visualización del resultado:

El Algoritmo 7 – Visualización al final de este anexo recibe el valor RMS y el vector de comparación, a partir del cual muestra el resultado de la medición, a través de uno de cuatro estados posibles.

Almacenamiento de los resultados:
Esta función consiste en llenar un vector de datos con los resultados obtenidos para cada eje, el almacenamiento de los datos se podría realizar en un dispositivo que tenga memoria EEPROM o cualquier otro medio de almacenamiento permanente, para este caso no se genera un algoritmo pues la rutina depende directamente del dispositivo elegido.

Conexión con el PC:
La conexión con el PC debe seguir el esquema mostrado en la etapa de concepción del sistema, se deben generar una serie de banderas o ACK (acknowledgement) de modo que sea posible saber si existe comunicación de ambas partes, el algoritmo es el mismo que se propone para la conexión con el punto de medición, por lo cual se obvia su descripción.

Registro y cálculo de parámetros de adquisición y comparación:
Este módulo consiste en crear una tabla característica con los datos que se reciben desde el PC, utiliza el mismo algoritmo de registro de datos, pero con los valores de adquisición, de igual forma las rutinas de registro dependen del hardware de almacenamiento seleccionado.

Almacenamiento de tabla de parámetros:
Este almacenamiento cumple la misma función del módulo Almacenamiento de resultados, la cual consiste en grabar datos de manera permanente de modo que no se pierdan una vez se desconecta la alimentación del sistema y como se mencionó en ese módulo, la rutina depende directamente de las instrucciones

específicas del módulo elegido por lo cual no se genera un algoritmo en pseudocódigo.

Enviar Mediciones almacenadas:
El envío de datos al PC dependerá igualmente de las características de hardware elegido, puesto que normalmente este tipo de sistemas ya utilizan estándares tales como transferencia de archivos utilizados en los sistemas Windows, los cuales son compatibles entre todos los medios de almacenamiento, debido a que esto es un estándar, no se genera algoritmo en pseudocódigo.

Borrar Mediciones almacenadas en el sistema:
Este algoritmo consiste en borrar los sectores de memoria donde se encuentran almacenados los datos de los resultados del sistema, su programación dependen del protocolo que requiera el hardware especifico que almacena estos datos, por lo cual no se genera un algoritmo en pseudocódigo para esta rutina.

Funciones del punto de medición

Conexión con sistema central:
Del diagrama de conexión del punto de medición con el sistema central realizado en la etapa anterior, se genera el algoritmo para realizar el intercambio de datos entre esos sistemas, dicho algoritmo reúne las características principales del sistema de comunicación que se quiere implementar, de igual forma la conformación de las tramas y demás aspectos reales a implementar, dependen específicamente del protocolo de comunicación que se elija.

Adquisición de datos:
El algoritmo para la adquisición de datos, consiste en configurar el módulo de adquisición con los parámetros adquiridos en etapas posteriores a la ejecución de esta función, además, es el que administra y transforma los datos capturados, con

base en los diagramas mostrados en el documento de diseño conceptual realizado en la etapa anterior, Debido a que consiste en la escritura de parámetros de configuración sobre el módulo de adquisición elegido, no se genera algoritmo en pseudocódigo, sino que se deja para ser programado directamente sobre el prototipo.

Registro de datos:
El punto de medición debe realizar un registro temporal de los datos que adquiere del acelerómetro, esto con el fin de tener un respaldo de los datos mientras hace la transmisión, además del caso donde sea necesaria hacer una retransmisión de los datos, para esto el sistema simplemente construye un vector de datos donde almacena estos valores de modo que los pueda manipular, solo es necesario cerciorarse que exista la suficiente cantidad de memoria RAM para tener almacenado el vector sin que se desborde la pila del sistema.

DISEÑO DEL HARDWARE

Metodología Pahl&Beltz
Siguiendo el protocolo, el primer paso a realizar, es utilizar la metodología de Pahl&Beltz para evaluar los diferentes dispositivos con los cuales estará compuesto el sistema.

Paso 1: Tablas.
Las siguientes tablas contiene los tipos de arquitecturas que están en capacidad de cumplir la funcionalidad del módulo, se excluyen los componentes que el usuario ya tiene seleccionados tales como sensor y visualización, además de los componentes que solo poseen una forma de implementar, tal como el ADC.

Sistema central

Módulo	Forma 1	Forma 2	Forma 3
Controlador	Microcontrolador	DSP	FPGA
Almacenamiento	EEPROM	Flash	SD
Registro	RAM	Flash	
Comunicación	I2C	SPI	RS485

Punto de medición

Módulo	Forma 1	Forma 2	Forma 3
Controlador	Microcontrolador	DSP	FPGA
Registro	RAM		

Paso 2: Rutas.

A continuación se hace la elección de las rutas, no se hacen todas las combinaciones posibles debido al gran número de posibilidades de modo que no se toman en cuenta las combinaciones que a simple vista, no son muy funcionales o demasiados costosas.

Rutas Sistema central:

R1: Microcontrolador + EEPROM + RAM + I2C

R2: Microcontrolador + EEPROM + Flash + RS485

R3: Microcontrolador + Flash + RAM + SPI

R4: Microcontrolador + SD + RAM + SPI

R5: DSP + EEPROM + Flash + I2C

R6: DSP + Flash + RAM + I2C

R7: DSP + SD + Flash + RS485

R8: FPGA + EEPROM + RAM + SPI

R9: FPGA + Flash + RAM + RS485

Rutas Punto de medición:
R1: Microcontrolador + RAM
R2: DSP + RAM

Paso 3: Criterios de evaluación
Para este proyecto se necesita primordialmente los ítems de bajo costo y tiempo de desarrollo, sin dejar a un lado el desempeño, de este modo los criterios de evaluación serán:

Costo: 0.4
Tiempo de desarrollo: 0.3
Desempeño: 0.3

Dónde:
Valor de cero para el costo, equivale a que es muy caro y valor de 10, que es muy barato.
Valor de cero para tiempo de desarrollo, equivale a que es muy demorado y valor de 10, que es muy rápido.
Valor de cero para Desempeño, equivale a que tiene muy mal desempeño, y valor de 10, que tiene muy buen desempeño.

Paso 4: Valoración de las rutas.
Rutas sistema central
R1: Costo=8, Tiempo desarrollo=7, Desempeño=7
R2: Costo=8, Tiempo desarrollo=5, Desempeño=8
R3: Costo=8, Tiempo desarrollo=8, Desempeño=7
R4: Costo=7, Tiempo desarrollo=5, Desempeño=4
R5: Costo=6, Tiempo desarrollo=6, Desempeño=7
R6: Costo=5, Tiempo desarrollo=4, Desempeño=8
R7: Costo=4, Tiempo desarrollo=4, Desempeño=9

R8: Costo=2, Tiempo desarrollo=4, Desempeño=8
R9: Costo=2, Tiempo desarrollo=3, Desempeño=9

Rutas Punto de medición:
R1: Costo=7, Tiempo desarrollo=6, Desempeño=7
R2: Costo=4, Tiempo desarrollo=3, Desempeño=9

Paso 5: Evaluación de las rutas.
Rutas sistema central
R1: 8(0.4) + 7(0.3) + 7(0.3)= 7.4
R2: 8(0.4) + 5(0.3) + 8(0.3)= 7.1
R3: 8(0.4) + 8(0.3) + 7(0.3)= 7.7
R4: 7(0.4) + 5(0.3) + 4(0.3)= 5.5
R5: 6(0.4) + 6(0.3) + 7(0.3)= 6.3
R6: 5(0.4) + 4(0.3) + 8(0.3)= 5.6
R7: 4(0.4) + 4(0.3) + 9(0.3)= 5.5
R8: 2(0.4) + 4(0.3) + 8(0.3)= 4.4
R9: 2(0.4) + 3(0.3) + 9(0.3)= 4.4

Rutas Punto de medición:
R1: 7(0.4) + 6(0.3) + 7(0.3)= 6.7
R2: 4(0.4) + 3(0.3) + 9(0.3)= 5.2

Como se puede observar, luego de realizar la comparación, las tres primeras rutas del sistema central, cumplen de mejor manera los criterios de desarrollo del proyecto, así mismo, los componentes de la ruta uno del punto de medición, serán los elegidos para desarrollar dicho subsistema.

Parámetros mínimos de hardware

A continuación se hace la recopilación de las características necesarias de los módulos de hardware, de modo que puedan cumplir con los requerimientos necesarios para que el sistema funcione correctamente, a partir de estas características se hace más fácil realizar la evaluación y elección de los dispositivos, en este punto se sigue describiendo uno a uno los módulos del sistema, si bien existen elementos de hardware que están en capacidad de cumplir las funciones de diferentes módulos integrados en un solo componente, se deja dicha elección para cuando se haya hecho la evaluación y se elija completamente todo el hardware.

Comunicación entre los dispositivos:
La comunicación entre los dos sistemas debe ser confiable, rápida y eficiente, como parámetros mínimos, se necesita:

Característica	**Valor**
Velocidad de trasferencia	Mínimo 1Kbps
Características:	- Alta inmunidad al ruido - Conector fácilmente removible - Dos líneas de más para alimentación

Existen diferentes protocolos para comunicar dispositivos electrónicos, entre otros, se encuentran el I2C, SPI, RS232, RS485, etc. cada uno posee sus ventajas y desventajas, en la siguiente tabla se muestran algunas características de cada uno.

Nombre	Nro. de líneas	Características	Ventajas	Desventajas
I2C	3	Protocolo en Bus	-Sistema Maestro – Esclavo, lo que permite tener varios dispositivos a la vez - Direcciona por software el dispositivo a comunicar	- No se puede tener más de un maestro. - No es full dúplex - Se pueden generar colisiones durante la transmisión
SPI	4	Protocolo en Bus full dúplex	-Mayor velocidad de transmisión que con I2C -Implementación en hardware es extremadamente simple	-Consume más pines de cada chip que I²C -sólo funciona en las distancias cortas
RS232	9	Transmisión serial en voltaje	-Tolerante al ruido al ser transmisión diferencial	Necesita muchas líneas de comunicación para aplicar completamente el protocolo
RS485	2	Transmisión en corriente	-Alimentación única de +5V -Inmune al ruido por ser una señal de corriente	- No alcanza velocidades muy grandes a comparación de otros protocolos.

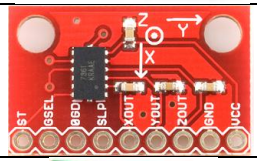
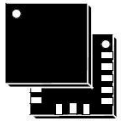
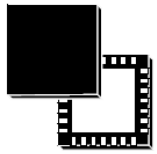
Si bien cualquiera de estos protocolos debe funcionar correctamente con el sistema a desarrollar, se dejara como método de elección el protocolo que mejor se acomode a los demás dispositivos seleccionados.

Hardware punto de medición

Sensor:
Según las especificaciones consignadas, se necesita un sensor con las siguientes características:

- Acelerómetro triaxial
- Acelerómetro de baja frecuencia
- Acelerómetros de bajo costo

En la siguiente tabla se encuentra una recopilación de algunos acelerómetros que poseen dichas características.

Referencia	Imagen	Rango	Interfaz	Ejes	Ancho de banda	Alimentación	Características	Costo
ADXL345		±2, 4, 8, 16g	SPI , I^2C	3	3200Hz	2.0-3.6V, 40-145µA	- Módulo ADC de 13 Bits integrado - Salida digital	US$30
MMA7361		±1.5g, ±6g	Análoga	3	400Hz	2.2V - 3.6 V	- Filtro implementado	US$20
LIS302		± 2g/± 8g	SPI, I^2C	3	400 Hz.	2.16 V - 3.6V	- Filtro embebidos - Sobrevive a caídas de 10000g	US$33
LIS3DH		±2*g*/±4*g*/ ±8*g*/±16*g*	SPI, I^2C 16 bit	3	1250Hz	1.71V - 3.6 V	-Módulo ADC de 16 Bits integrado -SIN base.	US$6
BMA020		±2*g*/±4*g*/ ±8*g*	SPI, I^2C 10 bit	3	1500Hz	2V - 3.6 V	Módulo ADC de 10 Bits integrado	US$6
LIS3LV02		±2g/±6g	SPI, I^2C 12 bit	3	2560Hz.	2.16V - 3.6V	Módulo ADC de 12 Bits integrado	US$13

Conversor análogo digital (ADC):

De la descripción realizada en la etapa de diseño conceptual, se toman las características principales que debe cumplir el ADC.

Característica	Valor
Resolución mínima	12 Bits
Velocidad mínima	2.5Ksps

Registro:

El sistema de registro de datos, tanto del punto de medición, como el del sistema central debe ser considerable, puesto que se van a capturar datos a 12 bits, lo que equivalen 2 bytes por cada dato, además, son 3 ejes y por cada eje, son 1024 datos, lo cual arroja que es necesario almacenar aproximadamente 6144 Bytes. O sea que mínimo se necesitan 7KB de almacenamiento.

Por otra parte el sistema de registro debe tener una velocidad de almacenamiento suficiente para que pueda guardar un dato, antes de que llegue el siguiente, por lo cual debe ser muy superior a la frecuencia de muestreo, por lo cual se propone un valor de 100us.

Característica	Valor
Capacidad de almacenamiento	Mínimo 7Kbytes
Velocidad de almacenamiento	Máximo 100uS

Controlador:

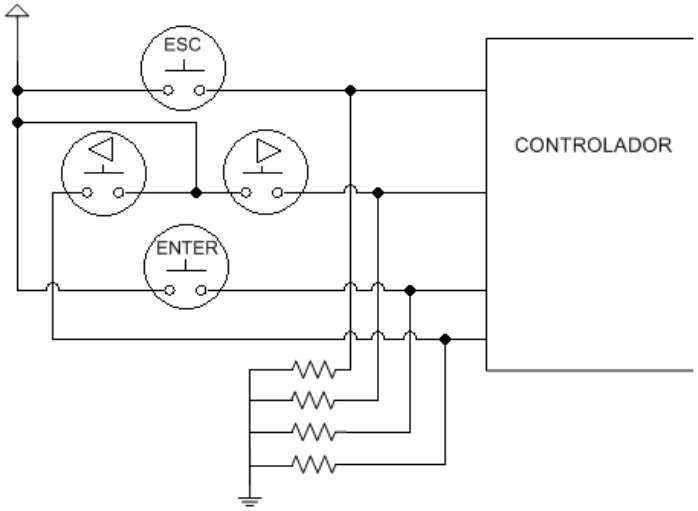
De las características descritas en la concepción del sistema, se hace la recopilación de las características de este componente, se necesita que tenga una buena velocidad de operación, además debe tener la suficiente cantidad de pines para poder tener conectados todas las comunicaciones, con el acelerómetro, con el registro y con el punto central.

Característica	Valor mínimo
Velocidad de CPU	5 MIPS
Bytes de RAM	4Kb
Mínimo Nro. Pines I/O	10

Hardware sistema central

Sistema de botones:

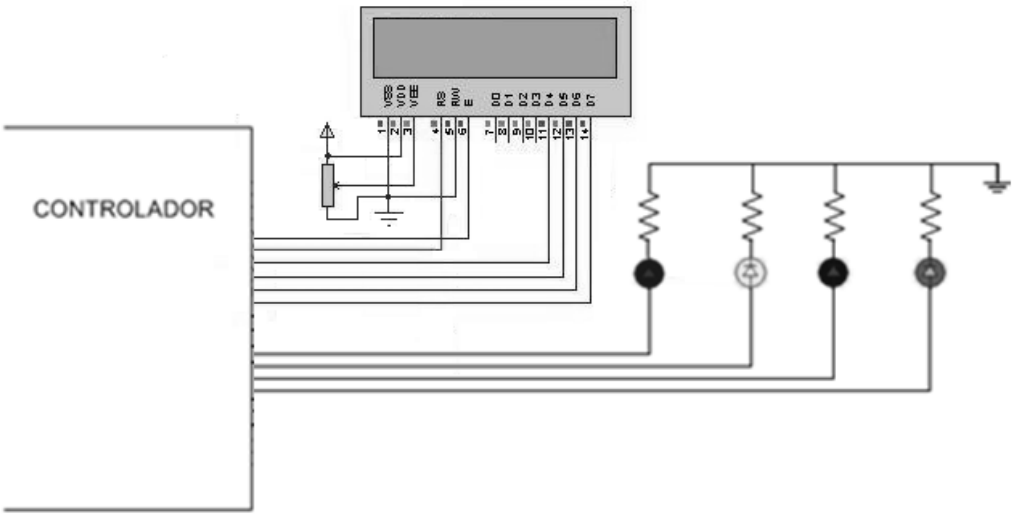
Este sistema no tiene ninguna complejidad y puede ser especificado inmediatamente, estos botones son implementados con resistencias Pull-Down, de modo que cuando los botones no estén presionados, las resistencias pondrán un valor de cero voltios en el puerto de microcontrolador, evitando así la lectura de valores aleatorios producidos por ruido electromagnético, el diagrama de este módulo es el siguiente:



Se observa que el voltaje en un puerto cambiara una vez es presionado el botón correspondiente, se debe tener en cuenta en la programación del software, la detección de la pulsación por interrupciones y la ejecución de rutinas de antirebote para contrarrestar las fluctuaciones de voltaje producidas por el sistema mecánico del botón.

Sistema de visualización:

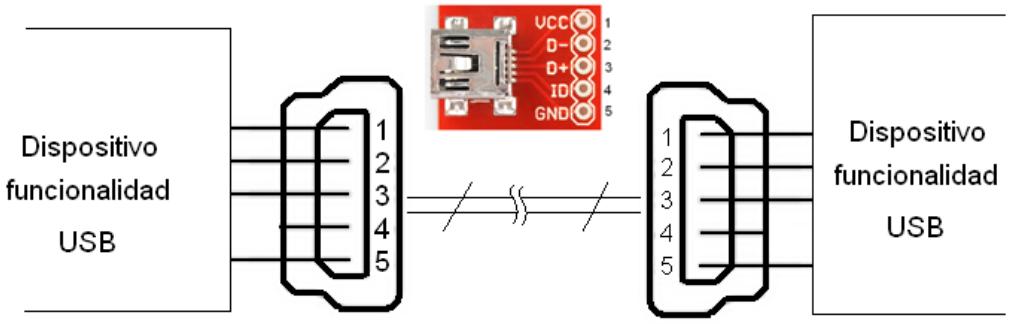
Este sistema tampoco posee mayor especificación dado que el usuario ya definió desde el inicio, el tipo de pantalla que quiere y la forma de visualización, la cual consiste en una pantalla alfanumérica de 2x16 y un sistema de 4 LEDs de diferente color para proporcionar una indicación un poco diferente del estado de la máquina, el diagrama de este hardware, se muestra a continuación.



Comunicación PC:

Puesto que el usuario ya había definido desde las especificaciones el tipo de comunicación a utilizar, se define que parte del estándar utilizar, para este caso

será el llamado USB CDC (USB communications device class) el cual genera un canal de comunicación virtual con el PC, a través del cual pueden ser implementadas interfaces de control y comunicación con un dispositivo externo al PC, este parte del estándar es muy utilizado en sistemas embebidos, tales como teléfonos celulares, máquinas fax y módems, existen dispositivos que cumplen las funcionalidades del hardware y librerías de software que brindan las herramientas para utilizar este protocolo, puesto que se busca que el sistema tenga un diseño pequeño se debe usar un conector mini USB como puerto de comunicación del sistema, el diagrama de conexión de este pin a dispositivo es:



Almacenamiento:

Para el almacenamiento de los datos, se necesita un dispositivo que pueda guardar datos de manera permanente, aun cuando no exista alimentación alguna, la capacidad de almacenamiento necesaria no es mucha, en la siguiente tabla se muestran los datos a almacenar, en segunda fila hay un ejemplo de una medición guardada y en la tercera se encuentra el tamaño de cada conjunto de datos.

Punto	**Fecha última medición**	**Vrms X [10^{-3}]**	**Vrms Y**	**Vrms Z**	**Estado máquina [A-D]**
D1	F24/11/10H10:51	1512	3245	1254	EA
1Byte	10Byte	2Byte	2Byte	2Byte	2Byte

De este modo se tiene que son necesarios 19Bytes de espacio de almacenamiento, por cada punto de medición, de modo que en una pequeña memoria de 1KB, se podrían almacenar los datos de 52 puntos de medición.

Registro:

El registro del sistema central cumpliría con la misma función del módulo del punto de medición, el cual es mantener los datos almacenados mientras se procesan, siguiendo los mismos cálculos realizados anteriormente, se obtiene que se necesita alrededor de 7KB de almacenamiento en memoria volátil, o sea que solo permanezca guardad mientras el sistema posea alimentación, las características de velocidad también deben ser las mismas, la tabla con estos valores es la siguiente:

Característica	**Valor**
Capacidad de almacenamiento	Mínimo 7Kbytes
Velocidad de almacenamiento	Máximo 100uS

Controlador:

El controlador del sistema central, será la pieza clave en el manejo de este dispositivo, será el encargado de procesar los datos y de coordinar el funcionamiento de todos los dispositivos periféricos, como se mencionó en la etapa de concepción del sistema, debe realizar también la supervisión del estado de carga de la batería, por lo cual deberá contar con un módulo de adquisición de datos del cual no es necesario tenga grandes prestaciones.

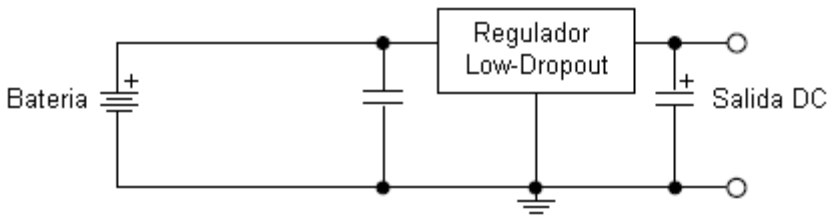
Para realizar las comunicaciones con los demás dispositivos (2 pines por dispositivo), mostrar mensajes en pantalla (7 pines) y atender los botones del usuario (4 pines), el controlador principal debe poseer una cantidad de mínimo 15 pines I/O (Entrada y Salida), a continuación en la siguiente tabla, se encuentran las características del controlador principal.

Característica	Valor mínimo
Velocidad de CPU	10 MIPS
Memoria de programa	32KBytes
Bytes de RAM	4Kb
Capacidad del canal ADC	10 bits
Numero de pines	40

Alimentación del sistema:

La alimentación del sistema es la que le brinda su portabilidad, debe buscar ser lo más eficiente posible, de modo que la batería que alimenta el sistema pueda lograr una mayor duración, se deben utilizar reguladores de voltaje LDO (*low-dropout*) y el control de encendido debe ser un swiche que desconecte totalmente la batería de modo que no tenga carga y siga consumiendo mientras que el sistema se encuentre apagado, para mayor flexibilidad del equipo, se van a utilizar un arreglo de baterías comerciales, las cuales deben ser recargables y además poseer el dispositivo cargador externo.

El circuito de esta etapa, va a depender de la necesidad de voltajes de los dispositivos elegidos, se propone una etapa reguladora para obtener los diferentes voltajes necesarios.

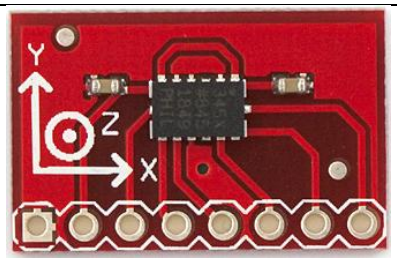


Elección de componentes

A continuación se hace la descripción de los módulos de hardware elegidos, inicialmente se hizo una investigación de los productos existentes en el mercado que estuvieran en capacidad de cumplir con las especiaciones descritas en las tablas de características mínimas de cada módulo, de igual forma fueron buscados los módulos de hardware que pudieran cumplir con la función de más de un módulo, dado que esto abarata el costo e integra diversas funciones que puede favorecer a un menor tiempo de desarrollo.

Módulos de hardware del punto de medición

El sistema elegido para el punto de medición es el acelerómetro ADXL345, el cual además de ser el elemento de medición, posee internamente una serie de filtros para el acondicionamiento de la señal, así como protocolo de comunicación nativo que permite la salida digital de los datos sensados, lo cual permite una mayor confiabilidad en las Mediciones, pues no existen valores análogos que puedan ser deformados con el ruido eléctrico y demás componentes de distorsión presentes en los diferentes medios, este acelerómetro posee el mayor ancho de banda de todos los acelerómetros descritos, lo que permite capturar un mayor rango de componentes de la señal medida, por otra parte, tiene la ventaja de tener una base con los conectores a sus pines, lo cual es una ventaja teniendo en cuenta que una mala soldadura puede generar errores en la señal medida, factor que se desea descartar, las características de este acelerómetro son:

ADXL345	Característica	Valor
	Numero de ejes	3: X,Y,Z
	Rango de medición	±2, 4, 8, 16g
	Ancho de banda	3200Hz
	Alimentación	2.0-3.6V
	Nro. bits ADC	13
	Comunicaciones digitales	SPI , I^2C
	Choque máximo para daño	10000 g

Módulos de hardware del sistema central.

Controlador

Como elemento principal del sistema central se elige el sistema de desarrollo llamado Mbed, el cual posee un microcontrolador ARM Cortex-M3, este sistema posee integrados los módulos de registro, almacenamiento y el RTC, además de tener un microcontrolador de muy buenas prestaciones que está en capacidad de realizar rápidamente la transformada de Fourier, este sistema posee además un circuito de acondicionamiento de voltaje para suministrar su propia alimentación y brindar una salida de 3.3 voltios para los dispositivos que lo requieren, como se puede apreciar, es un dispositivo muy completo y de mucha funcionalidad para este sistema.

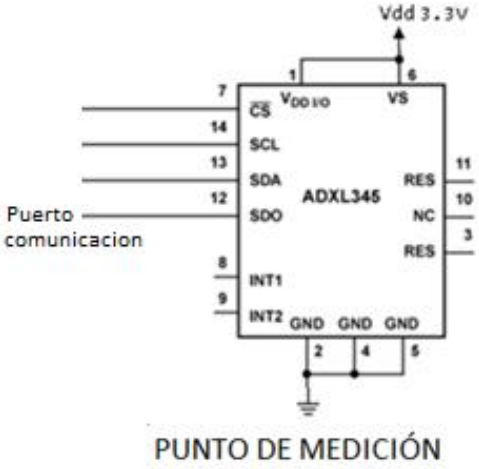
Mbed	**Característica**	**Valor**
	Microcontrolador	ARM Cortex-M3
	Arquitectura	32-bit
	Velocidad	96MHz
	Memoria Flash	512KB
	Memoria RAM	64KB
	Numero de pines	40-pin DIP
	Dimensiones	44mm x 26mm
	Interfaces:	• Ethernet • USB • CAN • SPI • I2C • Puertos I/O
	Funcionalidades	• GPIO • PWM • ADC • DAC • RTC

Comunicación entre dispositivos.

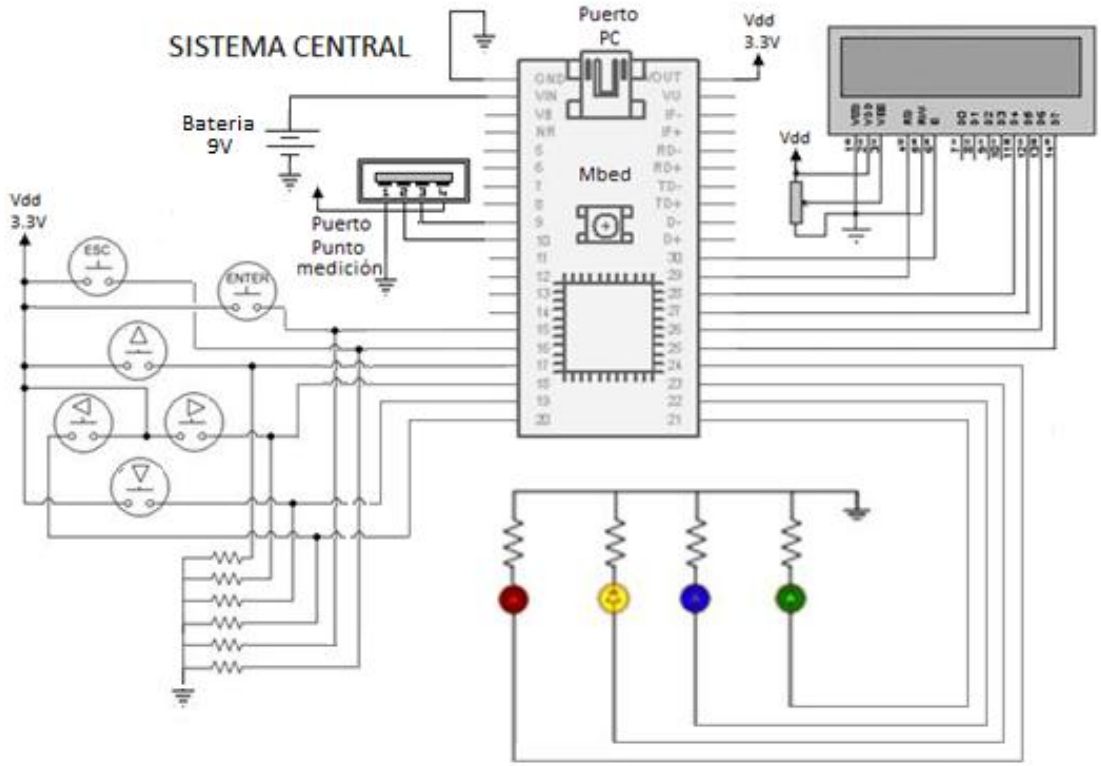
El sistema de comunicación entre el punto de medición y el sistema central, será a través de SPI, la cual presenta una buena velocidad de transmisión y es posible realizar corrección de Errores

De este modo el esquemático general del hardware del sistema es el que se muestra en la siguiente figura:

HARDWARE DEL SISTEMA



PUNTO DE MEDICIÓN



Se espera que con esta configuración de hardware sea posible cumplir con todos los requerimientos del sistema.

Elección del lenguaje y herramientas de programación.

En este caso el sistema Mbed posee una plataforma de desarrollo basada en web, la cual posee una gran cantidad de librerías y ejemplos que son de gran ayuda para la utilización de todas sus funcionalidades, el lenguaje de programación es C++.

Elección de la carcasa del sistema.

Con el fin de disminuir los costos que conlleva el diseño y fabricación de una carcasa personalizada, se opta por buscar en el mercado diferentes tipos de carcasas comerciales de modo que se acomode a la aplicación que se quiere desarrollar, luego de una intensa búsqueda, evaluación de costos, ventajas y desventajas de cada una, fueron elegidas dos carcasas que se muestran a continuación.

Carcasa del sistema central

Caja de plástico 152x83x33 mm de tres cuerpos y ventana para LCD de 2 pulgadas

Carcasa del punto de medición

Caja de aluminio con tapas de tamaño 53x44x12 mm

LISTA DE ALGORITMOS

**** Algoritmo 1 – Máquina de estados ****

**Este algoritmo muestra de manera general como es la máquina de estados que controla la ejecución de las funciones y la navegación de menús en el sistema, más que funcionalidad, provee la estructura a partir de la cual el sistema mantiene el orden de la ejecución, además de brindar al programador una secuencia y control de las rutinas ejecutadas.

Entradas:

Botones, sensor.

Salidas:

Pantalla, memoria, LEDs.

Variables:

Estado = 0; // Almacena el estado del sistema

Funciones:

Inicio

// Declaración de variables globales para todo el sistema

// Inicialización de los módulos del controlador.

Lazo infinito //La ejecución del programa se mantendrá en este lazo

//En cada repetición solo entrara en una sola rutina según el //valor de la variable

Si Eventos de interrupción

//Activación de Rutinas y/o variables de eventos

Fin si

Si *Estado* = 0 entonces

    //Funciones del estado 0

    //Estado = 1 //Asignando otro valor a la variable cambia de estado

Fin si

Si *Estado* = 1 entonces

    //Funciones del estado 1 //Cada estado tiene sus propios

    //Estado = 2 //Cambio de estado

Fin si

Si *Estado* = 2 entonces

    //función de ingresar parámetros //funciones propias

//Estado = 0 // se puede navegar fácilmente entre estados

Fin si

Si *Estado* = 3 entonces //No hay límite en el numero de estados

//función…

Fin si

Fin lazo infinito

Fin

**** Algoritmo 2 – Conexión con punto de medición****

**Este algoritmo contiene los pasos a realizar durante la conexión con el punto de **medición

Inicio

Entero Var1, Var2

Enviar "INI"

Mientras (Recibir Var1) diferente de "OK" //Espera la recepción

Recibir

Fin Para

Recibir identificador punto

Fin

**** Algoritmo 3 – Elección de la norma a utilizar ****

**Este algoritmo muestra cómo se hace la visualización y selección de la norma a utilizar, en donde la selección se hace con el botón *Enter* y el desplazamiento con las teclas laterales, y con base en la selección realizada en pantalla, el sistema toma una de las dos normas.

Entradas:

Botones; //Esta variable contiene el valor de la tecla presionada ej. Enter, Esc...

Salidas:

Pantalla; //En la pantalla del sistema mostrara los mensajes

Variables:

Estado = 0; // Almacena el estado del sistema

Norma=1; // Lleva la cuenta de cuál menú esta seleccionado

Constantes:

```
NroOpc =3;      // Constante que lleva el número de opciones
Inicio
Si Estado = 0 entonces                    //En este estado se muestran las normas
        Si Selección = 1 entonces         //dependiendo de la selección en que se
                Muestre: “Opción 1”       //encuentre, muestra esa norma
        Fin si
        Si Selección = 2 entonces
                Muestre: “Opción 2”
        Fin si
Estado =1.
Fin si
Si Estado = 1 entonces        //En este estado se espera la selección del usuario
        Si Botones = 0 entonces
Entrar el controlador a modo “Sleep”
        Si no entonces
                Si Botón = “>” entonces          //Desplazamiento hacia la derecha
                        Si Selección >= NroOpc
                                Selección = 1;   //Evita desbordes en la selección
                        Si no
                                Selección +1
                        Fin si
                        Estado =1
                Fin si
                Si Botón = “<” entonces          //Desplazamiento hacia la izquierda
                        Si Selección <=1
                                Selección = NroOpc;  //Evita desbordes en la selección
                        Si no
                                Selección -1
                        Fin si
                        Estado =1
                Fin si
                Si Botón = “Enter” entonces      //Va al estado según la selección
                        Caso Selección
```

Caso 1: *Norma*=0 // Estado para Opción 1

Caso 2: *Norma*=1 // Estado para Opción 2

Fin seleccionar

Estado =2

Fin si

Fin si

Fin si

Fin

**** Algoritmo 4 – Registro de datos ****

**

Inicio

// Recepción de Medición eje X

// Recepción de Medición eje Y

// Recepción de Medición eje Z

PARA(j=0; j< 2048*3; j++)

fprintf(Data,"%d\n",Datos[j]);

Fin

**** Algoritmo 5 – Cálculo del valor RMS de velocidad vibratoria ****

**

Inicio

Datos eje //datos de aceleración

Fmues //frecuencia de muestreo

Frec //Vector de frecuencias

Mag //Magnitud FFT

J,four, Rm, Acum //variables auxiliares

Fourier=FFT(Datos) // se aplica la transformada rápida de Fourier sobre los datos de aceleración

Mag=maginutud (Fourier) //se calcula su magnitud

PARA(j=0; j< 1024; j++)

```
        Frec[j]=(0:Fmues*2/Ndat:Fmues)
FINPARA
PARA(j=1;j<512;j++)
        Rm = (Mag[j]/Frec[j]);
        Rm = Rm*Rm;
        Acum= Acum +Rm;
        Rm=0;
FINPARA
VRMS=sqrt(Acum*0.5);
Fin
```

**** Algoritmo 6– Aplicación de la norma ****

** Algoritmo que recibe el valor RMS de velocidad vibratoria y devuelve la zona en la que se encuentra la máquina**

```
Inicio
// Declaración de variables de contadores
// Recepción del valor RMS
ISO108G1F[3] = {3.5, 7.1, 11};
ISO108G2F[3] = {2.3, 4.5, 7.1};
ISO108G3F[3] = {3.5, 7.1, 11};
ISO108G4F[3] = {2.3, 4.5, 7.1};
ISO108G1R[3] = {2.3, 4.5, 7.1};
ISO108G2R[3] = {1.4, 2.8, 4.5};
ISO108G3R[3] = {2.3, 4.5, 7.1};
ISO108G4R[3] = {1.4, 2.8, 4.5};
VecRef, VRMS;
        Si (Soporte==1)//SOPORTE FLEXIBLE
                Caso (Clase)
                        Caso 1:
                                Para (i=0;i<=2;i++)
                                VecRef[i]= ISO108G1F[i];
                                Fin Para
```

```
                    Fin caso
                    Caso 2:
                                Para (i=0;i<=2;i++)
                                VecRef[i]= ISO108G2F[i];
                                Fin Para
                    Fin caso
                    Caso 3:
                                Para (i=0;i<=2;i++)
                                VecRef[i]= ISO108G3F[i];
                                Fin Para
                    Fin caso
                    Caso 4:
                                Para (i=0;i<=2;i++)
                                VecRef[i]= ISO108G4F[i];
                                Fin Para
                    Fin caso
Sino    //SOPORTE RIGIDO
        Caso (Clase)
                    Caso 1:
                                Para (i=0;i<=2;i++)
                                VecRef[i]= ISO108G1R[i];
                                Fin Para
                    Fin caso
                    Caso 2:
                                Para (i=0;i<=2;i++)
                                VecRef[i]= ISO108G2R[i];
                                Fin Para
                    Fin caso
                    Caso 3:
                                Para (i=0;i<=2;i++)
                                VecRef[i]= ISO108G3R[i];
                                Fin Para
                    Fin caso
```

Caso 4:

Para (i=0;i<=2;i++)

VecRef[i]= ISO108G4R[i];

Fin Para

Fin caso

Fin si

Fin

**** Algoritmo 7 – Visualización ****

**

Inicio

// Recepción del valor RMS y el vector de referencia VecRef[]

Si (VRMS > VecRef[2])

Mostrar("ESTADO DE LA MÁQUINA: D \n");

Sino

Si (VRMS > VecRef[1])

Mostrar("ESTADO DE LA MÁQUINA: C \n");

Sino

Si (VRMS > VecRef[0])

Mostrar("ESTADO DE LA MÁQUINA: B \n");

Sino

Mostrar("ESTADO DE LA MÁQUINA: A \n");

Fin si

Fin si

Fin si

Fin

ANEXO G. CODIFICACIÓN Y ANÁLISIS DE RENDIMIENTO

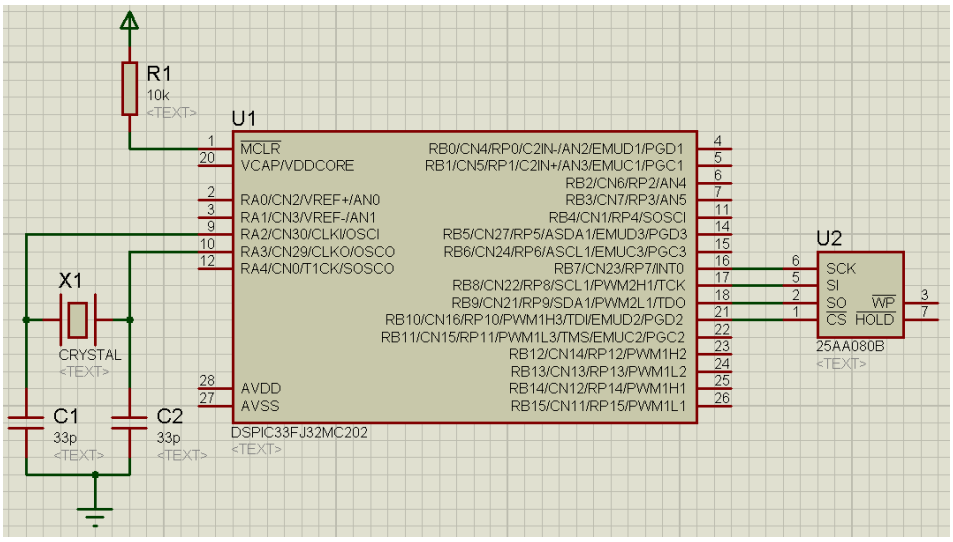
En esta etapa se busca corroborar que los diferentes dispositivos elegidos si poseen la total capacidad de cumplir con las actividades necesarias para las que fueron elegidos, para esto se inicia seleccionando una herramienta de simulación en la cual se realizan ensayos virtuales de algunas funciones del sistema, tales como la comunicación, y la transformada rápida de Fourier, que en este caso, son los dos aspectos más importantes de este sistema, a continuación se muestra el procedimiento realizado.

Elección de la herramienta de simulación

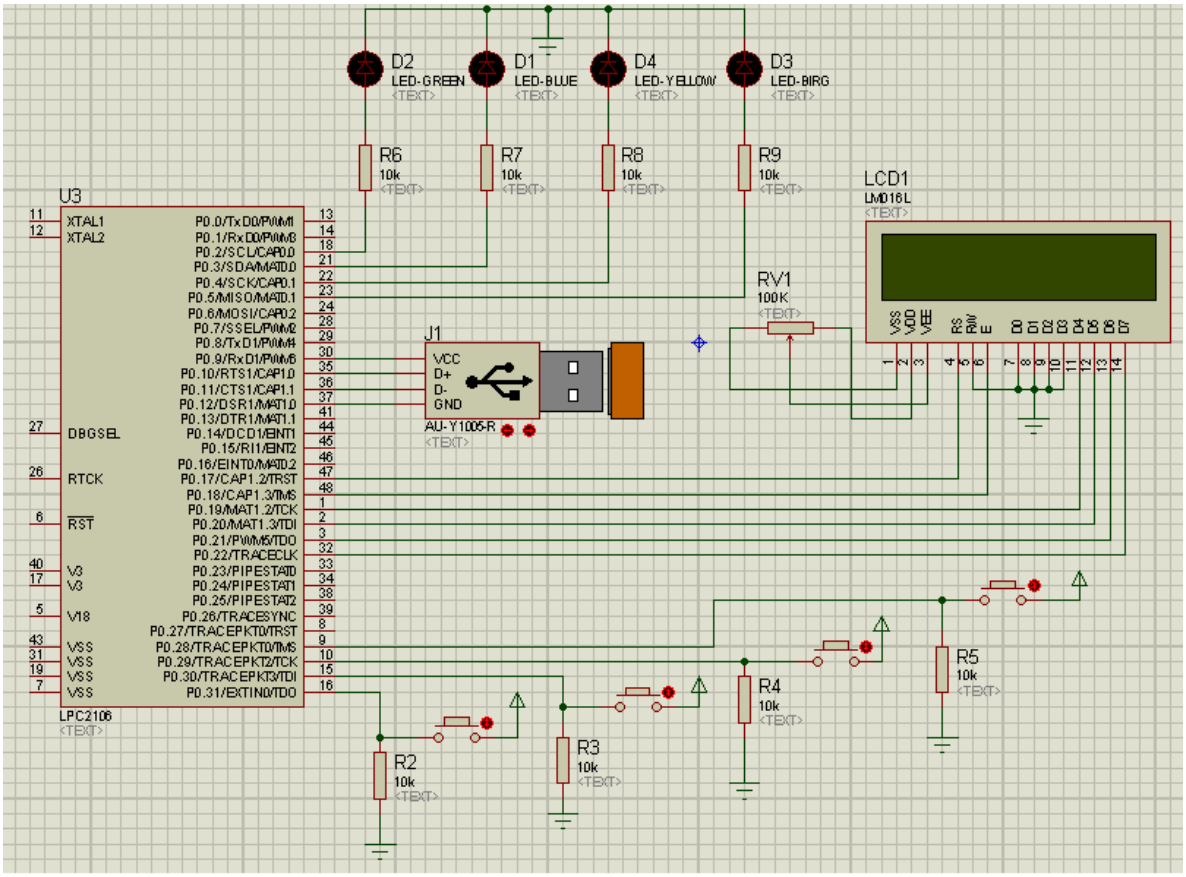
Para realizar las pruebas iniciales del sistema, se elige la herramienta de simulación virtual PROTEUS 7, la cual brinda un entorno grafico para el diseño de los esquemas electrónicos y permite la ejecución de programas de microcontroladores por medio de la carga de los archivos .HEX generados en algún compilador.

Puesto que para este sistema, los puntos más críticos se encuentran en la transmisión de datos y en la obtención del valor RMS de la velocidad vibratoria para el cual se necesita realizar la transformada de Fourier, se generan algunos circuitos y rutinas de código de prueba para comprobar su correcto funcionamiento en los dispositivos elegidos, el esquema de los circuitos simulados se muestra a continuación:

Esquemático simulación punto de medición:



Esquemático simulación Sistema central



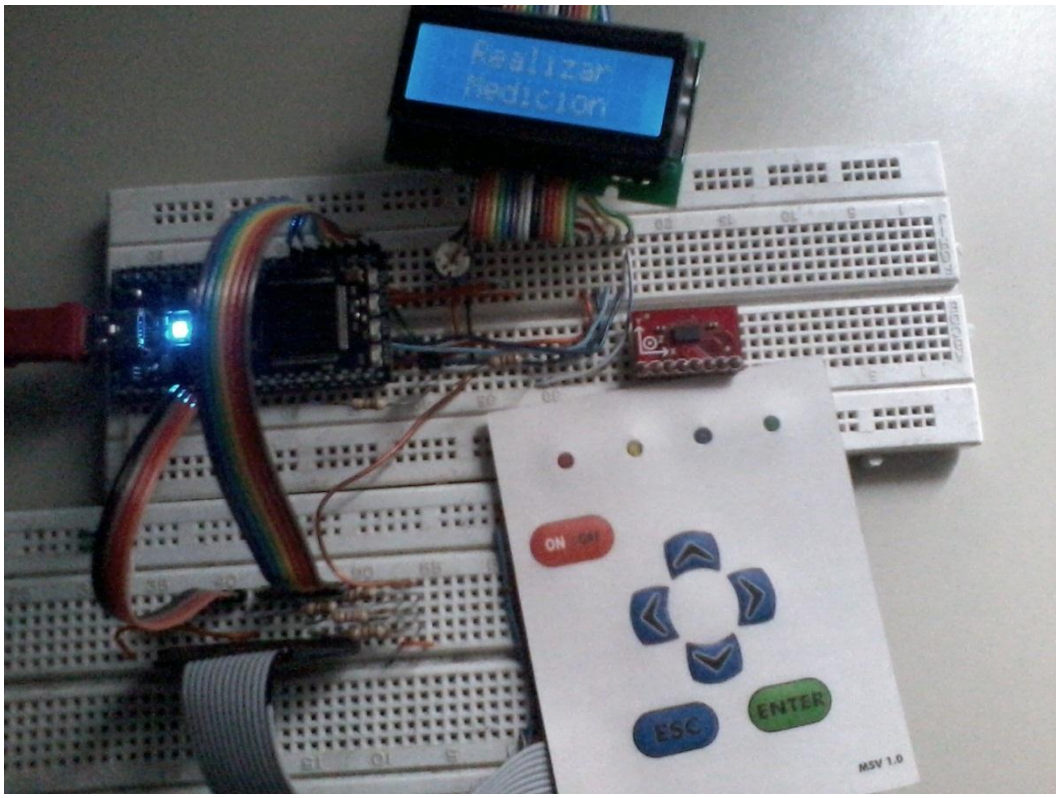
Las pruebas realizadas fueron los protocolos de comunicación SPI para probar la comunicación del Microcontrolador del punto de medición, con el acelerómetro ADXL345, se realizaron también pruebas del protocolo de comunicación USB, y además se generó la transformada de Fourier de 1024 datos aleatorios en el sistema central, para hacer estas pruebas fueron utilizados las funciones del entorno de programación MikroC de la empresa Mikroelektronika, las cuales permiten utilizar estos protocolos de comunicación sin necesidad de manejar todas las tramas manualmente, de igual forma se utilizó la función FFT(), la cual retorna las componentes reales e imaginaria de los datos y cuyo principio de funcionamiento está basado en la transformada Z.

Puesto que la documentación de esas funciones es bastante amplia en la página del fabricante, se omite la presentación de ellas en este documento, las pruebas realizadas en esta etapa, arrojaron resultados satisfactorios, lo cual permite continuar con la siguiente etapa del protocolo de desarrollo.

ANEXO H. IMPLEMENTACIÓN DEL HARDWARE

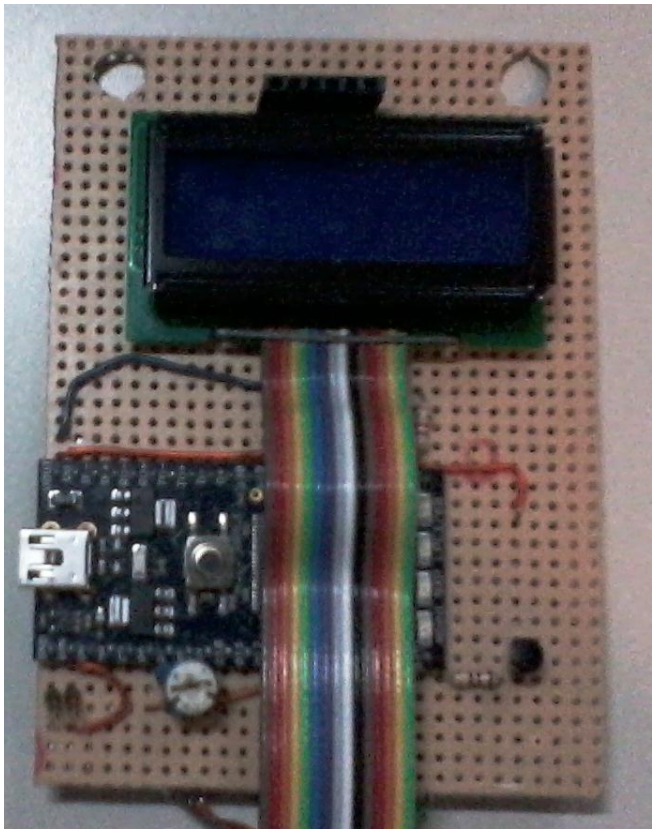
Una vez comprobado en simulación la correcta operación del hardware diseñado, se realizan algunas pruebas físicas de hardware para descartar algún problema físico no detectado hasta este punto, como se eligió una plataforma de desarrollo para realizar algunas tareas del sistema y puesto que casi ninguna plataforma de desarrollo posee herramientas de simulación virtual que permita poner a prueba algunas de sus características, es necesario realizar algunos ensayos con el dispositivo real.

Inicialmente se realizaron montajes en boards de prototipaje, en donde se evaluaba el correcto funcionamiento de los dispositivos de control, ya que normalmente las herramientas de simulación virtual no requieren de los componentes primarios para que el controlador funcione, tales como cristales y capacitores de polarización, de este modo se conocen cuáles son los componentes necesarios que deben estar incluidos en las tarjetas a fabricar.

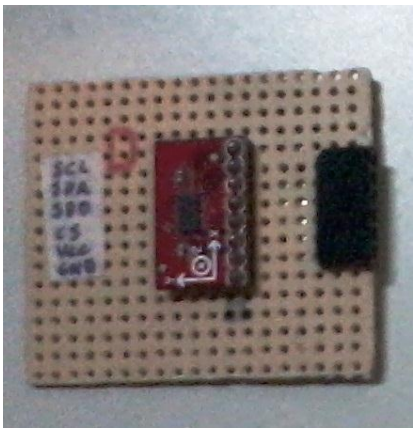


Posterior al funcionamiento en board de prototipaje, se procedió a realizar el montaje y soldadura de los circuitos electrónicos sobre boards universales, buscando obtener un hardware más estable y con mayores facilidades de manipulación

Sistema central



Punto de medición

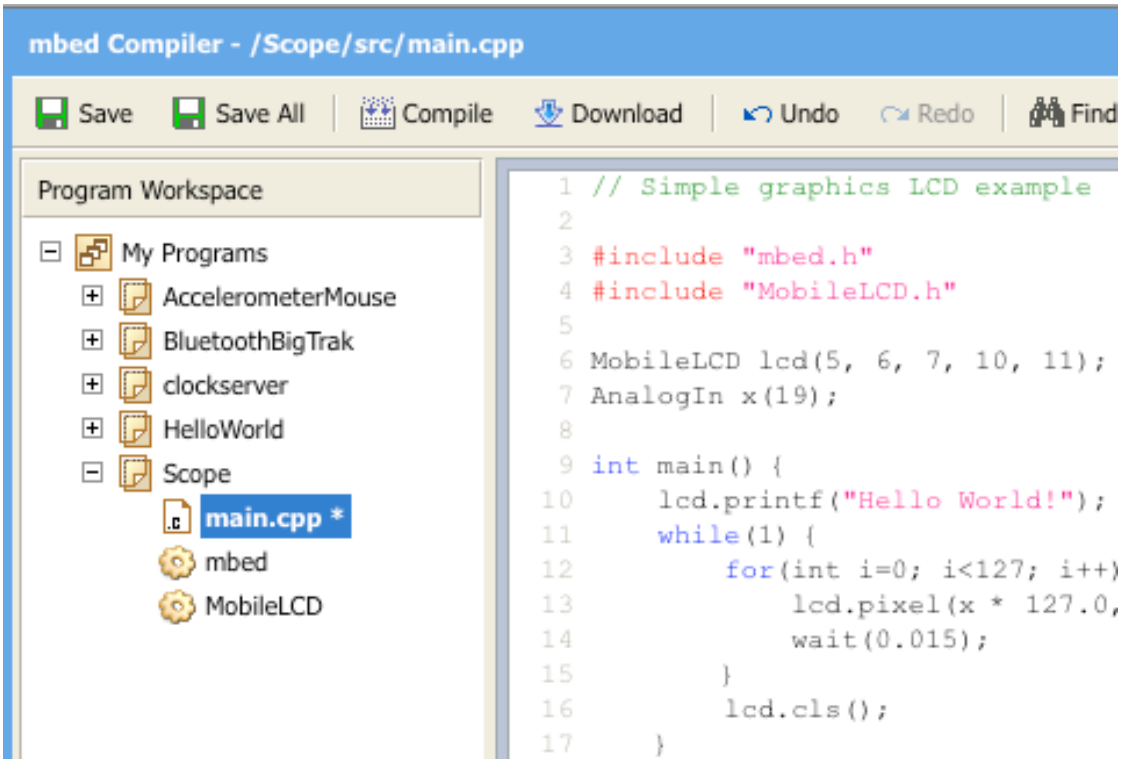


Todas estas tarjetas fueron diseñadas de modo que fueran compatibles con el tamaño y ubicación en la carcasa utilizada, de igual forma se fabricó un teclado de membrana que fue diseñado específicamente para la carcasa elegida, una vez ensamblado el sistema sobre la carcasa, tiene la siguiente apariencia.

Luego de tener todo el hardware armado y funcionando, se realizan las pruebas de funcionamiento de todos los módulos, para estas pruebas, se utilizaron las mismas rutinas de código usadas en las pruebas de los circuitos virtuales, y se comprobó correctamente que todo el hardware del sistema estaba operativo y funcional, gracias a esto es posible continuar con la siguiente etapa del desarrollo del sistema.

Siguiendo todas las recomendaciones del protocolo, se inicia con la programación del sistema a partir de los algoritmos generados en el etapa 4 – diseño detallado del sistema, inicialmente se generó la máquina de estados del sistema, que permitía la ejecución de segmentos de código controladamente, lo cual permite llevar un flujo de programa ordenado y evita posibles mal funcionamientos del sistema por entrar en estados desconocidos.

Como se definió previamente, para la programación del sistema central, se utilizó el entorno de programación propio del sistema Mbed, el cual está basado en la Web y que posee una serie de librerías que ponen a disposición una gran cantidad de funciones y aplicaciones, dicho entorno se muestra en la siguiente figura.



Una vez generada la máquina de estados del sistema, se procedió a programar el sistema de menús diseñado en el ANEXO C – concepción del sistema, y a partir de este, se fue dando funcionalidad a cada uno de los menús y módulos del sistema. Al final de este anexó se encuentran los códigos fuentes de este sistema.

```
/*********************PROGRAMA SISTEMA CENTRAL**********************
* Sistema que permite la evaluacion de la severidad vibratoria
 * a traves de la norma ISO 10816-3, donbde los parametros y los
 * resultados se almacenan y  se envían al PC por USB
*********************************************************************/
// Definicion de librerias //
#include <stdio.h>
#include <linux/rtc.h>
#include <sys/ioctl.h>
#include <sys/time.h>
#include <sys/types.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <termios.h>
#include <math.h>
#include "definiciones.h"

//////////DEFINICIONES PARA EL PRE-PROCESADOR///////////////
//Configuracion por defecto
#define DEFAULT_FS                  1000        //FRECUENCIA MUESTREO
#define DEFAULT_NMUES               1024        //NRO MUESTRAS
#define      DEFAULT_ACEL           660         //ADXL233
//Configuracion del hardware
#define BAUDRATE                    B9600               //Rata de baudios
#define MODEMDEVICE                 "/dev/ttyS0"        //Puerto serial B
#define      GPIO_PIN_DEVICE_BUT2 "/dev/gpio/69" //Boton 2
#define      GPIO_PIN_DEVICE_LED        "/dev/gpio/49" //LED 2
```

```
////////////////////////////////////////////////////////
//////////////////////VARIABLES GLOBALES//////////////////////////
//Variables para el hardware//
struct termios oldtio,newtio;       //configuracion serial
int fd_boton1;                          //boton 1
int fd_led;                             //led 2
int fd_serial;                         //puerto serie
//Variables para el manejo de ficheros//
FILE *Conf;             //Archivo con los parametros del usuario
FILE *Data;             //Datos recibidos de la Nexys
FILE *Ax;             //Datos de aceleracion en X con magnitud g
FILE *Ay;             //Datos de aceleracion en Y con magnitud g
FILE *fft;            //Datos de componentes de frecuencia en X
FILE *VecF;             //Datos de las frecuencias de la FFT
//Constantes para la comparacion de la norma//
float const ISO108G1F[3] = {3.5, 7.1, 11};
float const ISO108G2F[3] = {2.3, 4.5, 7.1};
float const ISO108G3F[3] = {3.5, 7.1, 11};
float const ISO108G4F[3] = {2.3, 4.5, 7.1};
float const ISO108G1R[3] = {2.3, 4.5, 7.1};
float const ISO108G2R[3] = {1.4, 2.8, 4.5};
float const ISO108G3R[3] = {2.3, 4.5, 7.1};
float const ISO108G4R[3] = {1.4, 2.8, 4.5};
//variables globales para la captura//
unsigned long fs;             //Frecuencia de muestreo
unsigned long Nmues;       //Numero de muestras
unsigned int Soporte;       //Soporte de la máquina 1 Flexible 2 Rigido
unsigned int Clase;             //Clase de la máquina 1 a 4
unsigned long Sens;             //Sensibilidad sensor
//Datos
int Datos[4096];
double VRe[1024];
double VIm[1024];
double FRe[1024];
double FIm[1024];
double Mag[1024];
double Frec[1024];
```

```
//////////////////////////////////////////////////////////////
//////////////////////FUNCIONES DE FOURIER//////////////////////////
//Fs(In_Re,In_Im,Nro Datos,)
void FourierShift(double *a_re, double *a_im, int l, double v)
{
   int k;
   double phi0 = v*2.0*M_PI/l;
   double s,c;
   double buffer_re, buffer_im;
   for(k=1; k<l; ++k)// k==0 skipped
   {
      c = cos(phi0*k);
      s = sin(phi0*k);
      buffer_re= (a_re[k]*c)-(a_im[k]*s);
      buffer_im= (a_re[k]*s)+(a_im[k]*c);
      a_re[k]=buffer_re;
      a_im[k]=buffer_im;
   }
}
//fft(Real_In, IM_In, Nro Datos,Real_Out,Im_Out,is)
void FastFourier(double *a_re, double *a_im, int n, double *x_re, double *x_im, int is)
{
      int j;
      int nh;

      if ( n==1 )  //valores de frecuencia cero.
   {
    x_re[0] = a_re[0];
    x_im[0] = a_im[0];
    return;
   }

   nh = n/2;
    double b_re[nh], c_re[nh], ev_re[nh], od_re[nh];
   double b_im[nh], c_im[nh], ev_im[nh], od_im[nh];
   for(j=0; j<nh;j++)
   {
```

```
      ev_re[j] = a_re[(2*j)];
      ev_im[j] = a_im[(2*j)];
      od_re[j] = a_re[(2*j)+1];
      od_im[j] = a_im[(2*j)+1];
    }
    FastFourier(ev_re, ev_im, nh, b_re, b_im, is);//pares
    FastFourier(od_re, od_im, nh, c_re, c_im, is);//impares
    FourierShift(c_re, c_im, nh, is*0.5);
    for(j=0; j<nh; j++)
    {
      x_re[j]    = b_re[j] + c_re[j];//salida, la suma p+i
      x_im[j]    = b_im[j] + c_im[j];
      x_re[j+nh] = b_re[j] - c_re[j];//espejo
      x_im[j+nh] = b_im[j] - c_im[j];
    }
}
////////////////////////////////////////////////////////////////
int main(void)
{
//////////////////////VARIABLES LOCALES//////////////////////////
      //Variables de conteo
      int i;
      long j;
      //long l,m;
      //Lectura de configuracion
      char cadena[30];
      char cadena2[60];
      long faux;                  //Velocidad de rotacion usuario
      long valor;                  //variable Auxiliar para la lectura
      unsigned long Ps;       //Periodo de muestreo para la FPGA
       //Envio de datos
      unsigned long int MAux;
      unsigned long int PAux;
      unsigned char dato;
      //Almacenamiento datos
      unsigned int Aux=0;
      float voltaje=0;
```

```
      //Valor RMS de vibracion
      double Acum;
      double Rm;
      double VRMS;
      //Comparacion con la norma
      float VecRef[3];
////////////////////////////////////////////////////////////

//Configuracion del módulo GPIO//
         system("modprobe gpio");        //Montar el módulo GPIO
         chdir("/dev/gpio");                  //Crear el nodo para el boton 1
         system("mknod 72 c 250 72"); // Abrir el nodo del boton 2
   fd_boton1 = open(GPIO_PIN_DEVICE_BUT2, O_RDWR );
      fd_led = open(GPIO_PIN_DEVICE_LED, O_RDWR );
         ioctl( fd_boton1, GPIO_CONFIG_AS_INP );       //Boton como entrada
      ioctl( fd_led, GPIO_CONFIG_AS_OUT );        //Led como salida
   printf("Configuracion Gpio.....OK\n");
////////////////////////////////////////////
//Configuracion del puerto serial (USB CDC)//
      // Abrir y configurar el puerto serie (USB CDC)
         fd_serial = open(MODEMDEVICE, O_RDWR | O_NOCTTY | O_NDELAY );
   if (fd_serial < 0)
    {perror(MODEMDEVICE); exit(-1);}
   else
    fcntl(fd_serial, F_SETFL, 0);
   //salva la configuracion actual del puerto serie
   tcgetattr(fd_serial,&oldtio);
   //limpia la estructura de la nueva configuaracion del puerto serie
   bzero(&newtio, sizeof(newtio));
   newtio.c_cflag = BAUDRATE  |PARODD |PARENB |CS8 | CLOCAL | CREAD  ;
   newtio.c_iflag = IGNPAR;
   newtio.c_oflag = 0;
   newtio.c_lflag = 0;
   newtio.c_cc[VTIME] = 0;
   newtio.c_cc[VMIN] = 1;
   tcflush(fd_serial, TCIFLUSH);
   // Asigna la nueva configuracion al puerto serie
```

```
   tcsetattr(fd_serial,TCSANOW,&newtio);
   printf("Configuracion Serial...OK\n");
////////////////////////////////////////////
//Leer la configuracion del archivo Conf.txt en USB//
      Conf = fopen ("/tmp/media/sda1/Parametros.txt", "r");
      if(Conf == NULL)
      {
       printf("No hay archivo de configuracion!!\n");
       printf("Tomando valores por defecto\n");
       fs=DEFAULT_FS;
       Nmues=DEFAULT_NMUES;
       Sens=DEFAULT_ACEL;
       Clase=1;
       Soporte=1;
      }
      else
   {
         //Lectura de la velocidad de operacion
         fscanf (Conf, "%[a-zA-Z]=%ld; #%[a-z A-Z 1-9,;]\n",cadena,&valor,cadena2);
       //dividir por 60 para obtenr HZ y mult x 2.5 para Nyquist
       fs= valor*25/600;
      Ps=12500000/faux; //12.5 MHz de la FPGA para el periodo de mues.
      printf("Frecuencia de muestreo: %ld\n",fs);
      printf("Periodo para FPGA: %ld\n",Ps);
      //Lectura del numero de muestras a tomar
      fscanf (Conf, "%[a-zA-Z]=%ld; #%[a-z A-Z 1-9,;]\n", cadena, &valor,cadena2);
      Nmues=valor;
      printf("No muestras: %ld\n",Nmues);
      //Lectura del tipo de soporte de la máquina
      fscanf (Conf, "%[a-zA-Z]=%ld; #%[a-z A-Z 1-9,;]\n", cadena, &valor,cadena2);
      Soporte=valor;
      printf("Tipo de soporte: %d\n",Soporte);
      //Lectura de la Clase de la máquina
      fscanf (Conf, "%[a-zA-Z]=%ld; #%[a-z A-Z 1-9,;]\n", cadena, &valor,cadena2);
      Clase=valor;
      printf("Clase: %d\n",Clase);
      //Lectura de la sensibilidad del acelerometro
```

```
        fscanf (Conf, "%[a-zA-Z]=%ld; #%[a-z A-Z 1-9,;]\n", cadena, &valor,cadena2);
        Sens=valor;
        printf("Sensibilidad: %ld\n",Sens);
       }
       fclose(Conf);
////////////////////////////////////////////////////////////
//Detiene el programa hasta que presionen el boton 2//
       printf("Presione Boton 2 para capturar \n");
       int lastinval,inval,ret_val;
       do {
                lastinval = inval;
                if( ( ret_val = ioctl( fd_boton1, GPIO_READ_PIN_VAL, &inval ) ) < 0 )
                {
                     perror( "ioctl: " GPIO_PIN_DEVICE_BUT2 );
                     close( fd_boton1 );
                     close( fd_led );
                     exit( EXIT_FAILURE );
                }
                usleep( 1000 );
           } while( !((lastinval == 1 ) && ( inval == 0 )) );
////////////////////////////////////////////////////////////////
//Envio de parametros y Recepción de datos//
       MAux= Nmues;
        PAux= Ps ;
   printf("Enviando Nro Muestras: %ld ; Periodo: %ld\n",MAux,PAux);
        // Enviar el numero de muestras a capturar
   for (i=1; i<= 3;i++)
     {
          if (i ==1)
       {
        dato = MAux;
        write(fd_serial, &dato, 1);
        printf(" %d NM\n",dato);
        }
       else
        {
              MAux = MAux >> 8;
```

```
        dato = MAux ;
        write(fd_serial, &dato, 1);
        printf(" %d NM\n",dato);
       }
     }
      // Enviar frecuecia de muestreo
   for (i=1;i<= 2;i++)
     {
      if (i ==1)
       {
            dato = PAux;
        write(fd_serial, &dato, 1);
        printf(" %d P\n",dato);
       }
      else
       {
        PAux = PAux >> 8;
        dato = PAux ;
        printf(" %d P\n",dato);
        write(fd_serial, &dato, 1);
        }
         }
   // Recepción de las muestras capturadas
   //Datos de 2 bytes, datos 2 ejes
   for ( j=0; j< Nmues*4; j++)
   dato = read(fd_serial, &Datos[j],1);
   // Indica que se han recibido las muestras
   printf("\nMuestras recibidas\n");
////////////////////////////////////////////////////
//Almacenamiento//
      printf("Abriendo archivos\n");
      //Abrir los ficheros
      Data= fopen("/tmp/media/sda1/Datos.txt","w+");
   if (Data == NULL)
   fprintf(stderr,"No se tiene acceso a los datos\n");
   Ax= fopen("/tmp/media/sda1/Ax.txt","w+");
   if (Ax == NULL)
```

```
   fprintf(stderr,"No se tiene acceso a la aceleracion X\n");
   Ay= fopen("/tmp/media/sda1/Ay.txt","w+");
   if (Ay == NULL)
   fprintf(stderr,"No se tiene acceso a la aceleracion Y\n");
       printf("Almacenando X\n");
      // Almacenamiento de las muestras en el fichero Nmues*4 datos
   for ( j=0; j< Nmues*4; j++)
                fprintf(Data,"%d\n",Datos[j]);
      // Almacenamiento de las muestras X en el fichero
   for (j= 0;j< Nmues*4;j = j+4)
    {
        Aux = Datos[j];
        Aux = Aux << 8;
        Aux = (Aux | Datos[j+1]);
        voltaje = Aux*3.3/4096;
        fprintf(Ax,"%f\n",(voltaje*1000-1650)/Sens);
        Aux = 0;
    }
        printf("Almacenando Y\n");
      // Almacenamiento de las muestras Y en el fichero
   for (j= 2;j< Nmues*4;j = j+4)
    {
     Aux = Datos[j];
     Aux = Aux << 8;
     Aux = (Aux | Datos[j+1]);
     voltaje = Aux*3.3/4096;
     fprintf(Ay,"%f\n",(voltaje*1000-1650)/Sens);
     Aux = 0;
    }
   fclose(Data);
       fclose(Ax);
       fclose(Ay);
   printf("Muestras almacenadas\n");
////////////////////////////////////////////////////////////
///Velcidad RMS para X///
       fft= fopen("/tmp/media/sda1/FFT.txt","w+");
       VecF=fopen("/tmp/media/sda1/Frec.txt","w+");
```

```
Ax = fopen("/tmp/media/sda1/Ax.txt","r+");
    sleep(1);
    for(j=0;j<Nmues;j++)
      fscanf(Ax,"%lf\n",&VRe[j]);
   VRe[0]= VRe[5];       //Descarto los 2 primero datos
   VRe[1]= VRe[9];
   //fft(In, #datos, Out,1)
   FastFourier(VRe,VIm,Nmues,FRe,FIm,1);
   //Magnitud de la fft
   for(j = 0; j < (Nmues/2); j++)
           Mag[j]=sqrt((FRe[j]*FRe[j])+(FIm[j]*FIm[j]));
   //Almacenamiento en archivo
for (j=0;j< (Nmues/2);j++)
      fprintf(fft,"%lf\n",Mag[j]);
   for(i=0;i<8;i++)
         printf("Re %f \n",VRe[i]);
    for(i=0;i<5;i++)
         printf("Im %f \n",VIm[i]);
   for(i=0;i<10;i++)
         printf("F %f \n",FRe[i]);
   for(i=0;i<10;i++)
         printf("M %f \n",Mag[i]);

    //Generacion del vector de frecuencias
   Frec[0]=0;                                  //Inicia valor en cero
   for (j=1;j< (Nmues/2);j++)
          Frec[j]=(Frec[j-1]+4.88281);  //intervalos de Fs/512 (2500/512)
   for (j=0;j< (Nmues/2);j++)
  {
  fprintf(VecF,"%lf\n",Frec[j]);
       Frec[j]=Frec[j]*6.2832;              //2 Pi
  }
   for(i=0;i<10;i++)
         printf("w %f \n",Frec[i]);
   Acum=0;
   for (j=1;j<(Nmues/2);j++)
  {
```

```
        Rm = (Mag[j]/Frec[j]);
        Rm = Rm*Rm;
        Acum= Acum +Rm;
        Rm=0;
    }
    VRMS=sqrt(Acum*0.5);
    printf("Vrms: %f \n",VRMS);
     fclose(Ax);
     fclose(VecF);
       fclose(fft);
////////////////////////////////////////////////////////////
//Comparacion con la norma//
      if (Soporte==1)//SOPORTE FLEXIBLE
      {
            switch (Clase)
            {
                  case 1:
                              for (i=0;i<=2;i++)
                              VecRef[i]= ISO108G1F[i];
                              break;
                  case 2:
                               for (i=0;i<=2;i++)
                              VecRef[i]= ISO108G2F[i];
                              break;
                  case 3:
                              for (i=0;i<=2;i++)
                              VecRef[i]= ISO108G3F[i];
                              break;
                  case 4:
                              for (i=0;i<=2;i++)
                              VecRef[i]= ISO108G4F[i];
                              break;
            }
      }
      else       //SOPORTE RIGIDO
      {
            switch (Clase)
```

```
        {
        case 1:
                        for (i=0;i<=2;i++)
                        VecRef[i]= ISO108G1R[i];
                        break;
              case 2:
                        for (i=0;i<=2;i++)
                        VecRef[i]= ISO108G2R[i];
                        break;
              case 3:
                        for (i=0;i<=2;i++)
                        VecRef[i]= ISO108G3R[i];
                        break;
              case 4:
                        for (i=0;i<=2;i++)
                        VecRef[i]= ISO108G4R[i];
                        break;
        }
}

if(VRMS > 15)
        printf("ERROR por mayor \n");
if(VRMS ==0)
        printf("ERROR por menor \n");

if(VRMS > VecRef[2])
        printf("ESTADO DE LA MÁQUINA: D \n");
else
{
        if(VRMS > VecRef[1])
        printf("ESTADO DE LA MÁQUINA: C \n");
        else
        {
              if(VRMS > VecRef[0])
              printf("ESTADO DE LA MÁQUINA: B \n");
              else
              printf("ESTADO DE LA MÁQUINA: A \n");
```

```
            }
        }
// Funcion de terminar///
   system("lxumount");
       printf("Memoria fuera\n");
        // Devuelve la configuracion inicial al puerto serie
        tcsetattr(fd_serial, TCSANOW, &oldtio);
        // cierra el puerto serie
        close(fd_serial);
////////////////////////////////////////////////////////////
return 0;
}
```

ANEXO J. PRUEBAS, DEPURACIÓN Y FINALIZACIÓN

Una vez se ha programado el sistema, se inician una serie de pruebas que confirman el correcto funcionamiento del producto, para esto se siguieron los siguientes procedimientos.

Pruebas en reposo.

Para esta prueba se dejó el sistema sobre una superficie firme y en reposo, de modo que el sistema no estaba sujeto a ningún tipo de vibración, de este modo se comprueba que al sistema estar midiendo únicamente la gravedad normal sobre el suelo, arroja un valor RMS de velocidad muy cercano a cero, debido que no se generaría ningún componente de frecuencia considerable, la respuesta del sistema a esta prueba siempre fue buena e indico que el estado de la medición era vibración tipo A, en esta prueba se realizó también la calibración del acelerómetro de modo que generara valores de lecturas en cero cuando estuviera sujeto a estas condiciones, quitándole así, los posibles offset de las Mediciónes de vibración.

Pruebas con vibraciones controladas

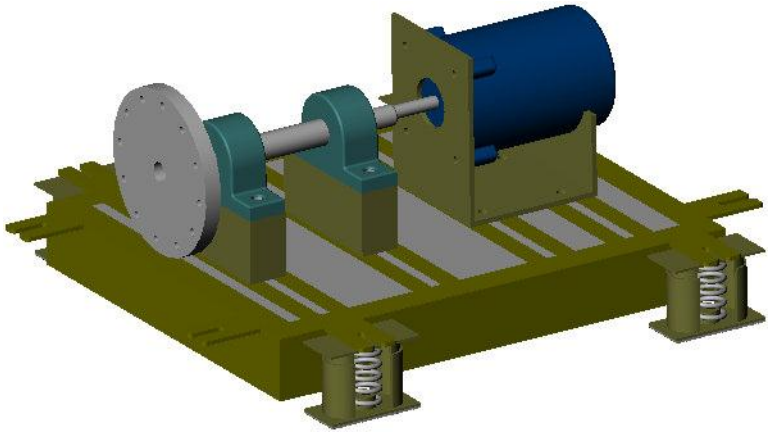
Esta serie de pruebas se realizaron con la ayuda de un excitador modal de la casa B&K de referencia Modal Exciter Type 4828, a partir del cual se generaron vibraciones a frecuencias y amplitudes conocidas.

Excitador modal B&K

Para esta prueba se hicieron una serie de mediciones en donde se calculaba manualmente el valor RMS de velocidad vibratoria teórico que debería arrojar la medición, en este caso, la prueba arrojo buenos resultados donde el rango de severidad vibratoria obtenido con el sistema siempre estuvo cercano al cálculo teórico de dicho valor, lo cual demuestra que las operaciones realizadas dentro del sistema embebido funcionan perfectamente.

Pruebas de medición en un banco de pruebas de desbalanceo.

Esta prueba consistió en hacer Mediciónes sobre un banco de pruebas para realizar desbalanceo sobre el eje de un motor, este banco consiste en un motor sobre una plataforma de cemento, la cual está aislada del suelo a través de un sistema de 4 resortes, teniendo así, un soporte flexible, sobre el eje del motor, se encuentra un volante en donde es posible adherir segmentos de masa que generar un desbalanceo de la máquina, aumentando así su severidad vibratoria, en especial cuando las frecuencias de vibración están cercanas a su frecuencia natural, la forma del banco de pruebas es la siguiente:



La prueba consistió en realizar inicialmente una medición con la máquina en funcionamiento normal y sin desbalanceo, el resultado de esta primera prueba fue correcto y se obtuvo un valor de vibración en A, luego se procedió a una serie de pruebas de medición, en donde cada vez se le agregaba una mayor cantidad de masa a la volante de desbalance, y se pudo corroborar que a

mayor desbalance, aumentaba más los valores RMS de velocidad vibratoria y por lo tanto el estado de la máquina.

Con las anteriores pruebas se comprueba que el sistema posee un funcionamiento óptimo y confiable, en esta etapa se realizó el ensamblaje total del sistema y se generaron los documentos del sistema, los cuales son:

- Hoja técnica del sistema
- Manual de usuario
- Informe de pruebas
- Documento con el procedimiento del chequeo del equipo

ANEXO K. EVALUACIÓN

Una vez finalizado el desarrollo del sistema, se hace la entrega oficial al grupo GEMI, quien es el que realiza la evaluación del sistema con base en el documento de especificaciones, puesto que en la etapa de pruebas, depuración y finalización se realizaron las pruebas necesarias y se revisó previamente que el sistema cumpliera con todos los objetivos y condiciones requeridas desde el principio, no se recibieron correcciones adicionales al sistema, de este modo se cumple con la entrega de un sistema funcional de bajo costo para la evaluación de la severidad vibratoria en máquina rotativas, de igual forma se valida el protocolo de desarrollo para el diseño y construcción de un prototipo, a continuación, se muestra la tabla de evaluación para el sistema entregado y el acta de finalización del proyecto.

TABLA DE EVALUACIÓN DEL SISTEMA DESARROLLADO

Nro.	Ítem	Cumple	
1	**Funcionalidad**	✓	X
1.1	El sistema debe tener capacidad de adquirir señales de vibración	✓	
1.2	El sistema debe calcular la severidad vibratoria de una máquina y realizar una clasificación según normas ISO 10816-3 o ISO 2372 según la elección del usuario	✓	
1.3	El sistema mostrará el resultado de la evaluación de la severidad a través de una de cuatro luces de diferente color que representan los estados de la severidad vibratoria según la norma elegida	✓	
1.4	A través de un programa en un PC, el sistema debe generar un reporte del estado de las máquinas	✓	
1.5	Cada punto de medición solo deberá funcionar cuando se conecta el sistema central, de modo que cuando no hay conexión con este, el dispositivo este totalmente desactivado	✓	
1.6	El sistema debe indicar cuando no existe comunicación con los dispositivos	✓	
2	**Composición**		
2.1	El sistema estará compuesto por 2 tipos de sistemas diferentes	✓	
2.1.1	Subsistema llamado punto de medición, en el cual se encuentra el transductor de aceleración triaxial, el acondicionamiento y la adquisición de los datos de vibración	✓	
2.1.2	Subsistema llamado sistema central que es donde se realiza el procesamiento de los datos y la visualización de los resultados	✓	
2.2	El sistema central deberá tener un método de visualización compuesto por 4 luces de colores para mostrar el estado de la máquina y una pantalla alfanumérica para el ingreso de los datos	✓	
2.3	El sistema central deberá contener mínimo cuatro botones a partir de los cuales se realizara la navegación de menús en pantalla y el control del sistema	✓	
2.4	El sistema completo deberá estar compuesto por 3 puntos de medición y un sistema central que evalúa la severidad vibratoria	✓	
2.6	Los puntos de medición deberán ser diseñados para ser montados permanentemente en la máquina a medir	✓	
2.7	El sistema deberá contar con una carcasa que lo proteja de las condiciones ambientales a las que estará sometido	✓	
2.8	El tipo de comunicación entre los puntos de medición y el sistema central, así como de este último con el PC, será cableada	✓	

Nro.	Ítem	Cumple	
3	**Parámetros de operación**	✓	X
3.1	El sistema deberá tener una resolución de adquisición de datos de mínimo 12 bits	✓	
3.2	El sistema deberá estar en capacidad de realizar Mediciónes en un rango entre 2 y 1000 Hz según como lo dicta la norma	✓	
3.3	El sistema deberá estar en capacidad de operar en temperatura ambientales ente 0 – 70°C	✓	
4	**Parámetros de sensores**		
4.1	Acelerómetro triaxial	✓	
4.2	Acelerómetro de baja frecuencia	✓	
4.3	Acelerómetros de bajo costo	✓	
4.4	Con rango dinámico menor a ±50 g y mayor a ±10g	✓	
5	**Características físicas**		
5.1	El sistema deberá tener una carcasa que lo proteja del medio ambiente, con preferencia de tener protección IP65	✓	
5.2	El sistema debe ser portable	✓	
5.3	El sistema debe tener baterías recargables	✓	
5.4	El sistema debe tener un peso inferior a 1 Kg	✓	
6	**Aspectos misceláneos Relevantes**		
6.1	Documentación completa del sistema	✓	

Concepto del evaluador:

_____

_____

_____

Firma del evaluador

ACTA DE ENTREGA DE PROYECTO

En la ciudad de…… a los…… días del mes de……. del año…….. se realiza la entrega final del proyecto ……. el cual tuvo una duración …….. meses, la entrega la realiza ……. Con CC…….. a …….. con CC……..... quien lo recibe previa constatación y verificación, Dejando constancia de la conformidad de la misma, para su efecto firman las partes correspondientes al pie de esta acta.

_____ _____

Firma quien recibe Firma quien entrega