Facultad 5

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas.

Sistema de monitoreo y control en tiempo real de un motor de corriente directa para prácticas de Laboratorio Docente.

Autor (es): Jimmy Bassa Martínez

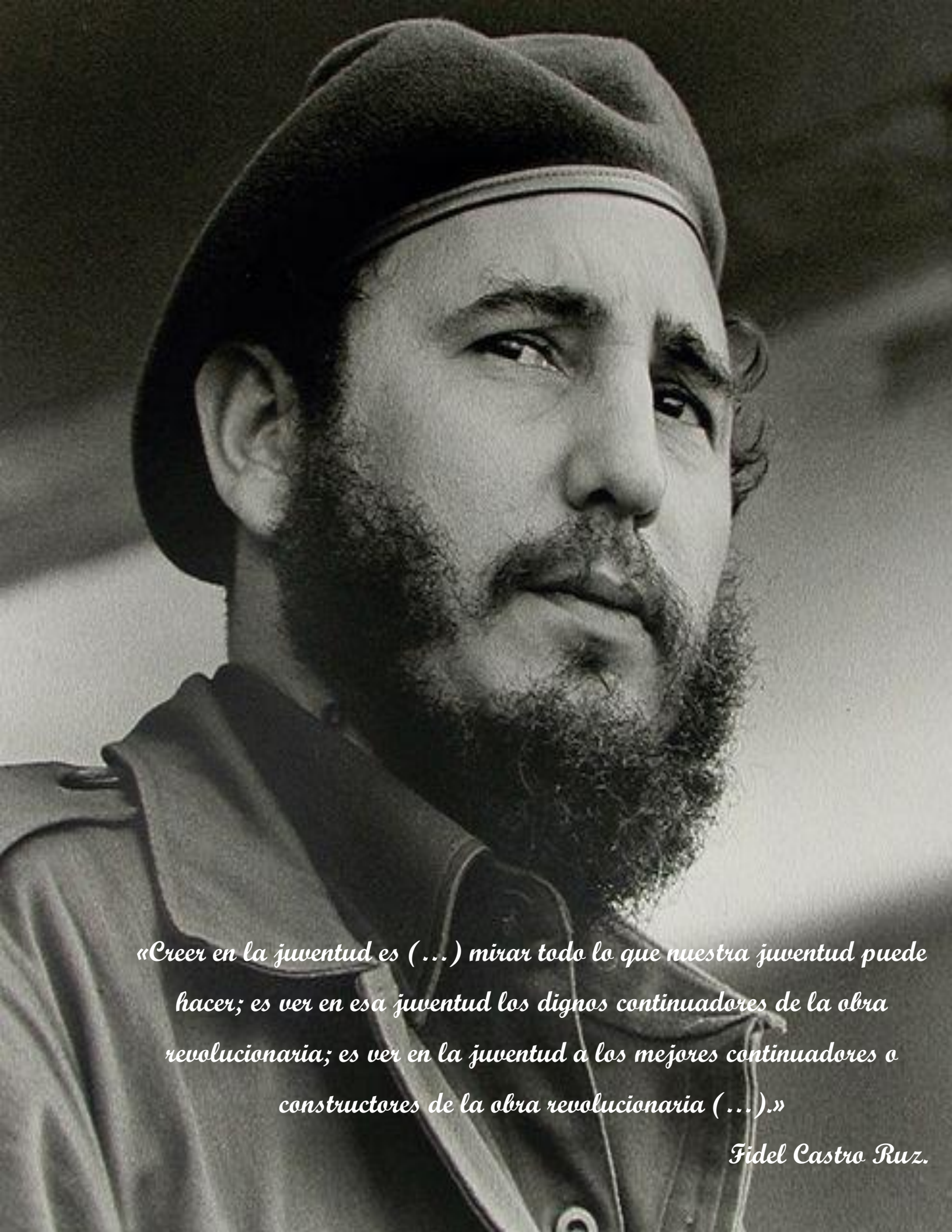
Tutor (es): Ing. Yoan Parra Nápoles

Ing. Frank Pacareu de la Cruz

Co-tutora: Ing. Claudia Larramendi Ferrás

La Habana, julio de 2014

"Año 55 de la Revolución"



«Creer en la juventud es (…) mirar todo lo que nuestra juventud puede hacer; es ver en esa juventud los dignos continuadores de la obra revolucionaria; es ver en la juventud a los mejores continuadores o constructores de la obra revolucionaria (…).»

Fidel Castro Ruz.

Declaración de autoría

Por este medio se declara que soy el único autor de este trabajo y se autoriza a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estime pertinente con este trabajo.

Para que así conste firmo la presente a los ____ días del mes de Julio del año 2014.

Jimmy Bassa Martínez

_____

Firma del Autor

Ing. Yoan Parra Nápoles

_____

Firma del Tutor

Ing. Frank Pacareu de la Cruz.

_____

Firma del Tutor

Datos de contacto

Nombre y Apellidos: Yoan Parra Nápoles

Institución: *Universidad de las Ciencias Informáticas (UCI)*

Título: *Ingeniero en Ciencias Informáticas*

E-mail: *yparra@uci.cu*

Facultad: *5*

Departamento: *Departamento de Tecnología*

Nombre y Apellidos: Frank Pacareu de la Cruz

Institución: *Instituto de Cibernética, Matemática y Física (ICIMAF)*

Título: *Ingeniero en Ciencias Informáticas*

E-mail: *frank@icimaf.cu*

Departamento: *Control Automático*

Nombre y Apellidos: Claudia B. Larramendi Ferras.

Institución: *Universidad de las Ciencias Informáticas (UCI)*

Título: *Ingeniera en Ciencias Informáticas*

E-mail: *cblarramendi@uci.cu*

Facultad: *5*

Departamento: *Soluciones SOA.*

Agradecimientos

Quiero agradecer a todas las personas que de una manera u otra han aportado su granito de arena para el desarrollo de este trabajo, a Yoan y Elizabeth por el apoyo en todo momento, por todas las horas que me dedicaron cuando necesitaba de gran ayuda, a Frank por estar siempre dispuesto a ayudar y por brindarnos siempre la atención adecuada cuando visitábamos el ICIMAF, a Claudinha por ser una persona que siempre está dispuesta a ayudar en lo que haga falta, de corazón muchísimas gracias por dedicar parte de tu tiempo.

A mi mamá por ser el ejemplo de mujer luchadora, por ser la mejor madre del mundo y por brindarme siempre su apoyo incondicional.

A mi papá por ser el ejemplo a seguir

A mis compañeros del grupo 5107 sin dejar de mencionar a ninguno que son los mejores compañeros de aula que he tenido en mi vida y nunca olvidaré los momentos felices que pasamos juntos en primer año, de ustedes llevaré por siempre el mejor de los recuerdos.

A mis hermanos de apartamento Roger, Henry, Marcos, Yosmany, Eduardo, Sandy, Yoandry (el corto), sin palabras nunca los olvidaré, gracias por compartir estos cinco años juntos, son los mejores del mundo.

Al piquete del futbol (el Nino, Rogelio, Jose, el Chaty, Medero, Oscarito, Yalbert, el Riki, Anniel (Musulungu), el Luiso, Miguel, el Mursu, el Yoko, al Yuma, a los de la fac2, fac 1, en fin a todos aquellos que de una manera u otra siempre contaron conmigo para gozar en las canchas de futbol), muchísimas gracias por tenerme presente.

A mis tres mujeres en la escuela (Beatriz, Yamila y Liz) que son las personas con las que más he convivido en la universidad y por soportarme durante estos cinco años, gracias por tenerme presente y compartir cada momento junto, nunca las olvidaré ya que es difícil olvidar a personas tan especiales como ustedes.

Agradecimientos

A mis amigos fuertes que se van conmigo a cada lado que vaya, porque estarán siempre en mi corazón (Hayri, Eliades, Alejo, Nino, Handy,Parra, Adrian,)

A las polillitas que más quiero en la escuela, Elizabeth, Anayansi, Ailyn y Ailen, muchas gracias por dedicar parte de sus tiempos cuando más lo necesitaba, nunca las olvidaré porque ustedes significan mucho para mí.

A mis amigos Prevot, Lester, Quiler y Marisel por brindarme su casa para poder trabajar hasta las tantas horas de la noche y por compartir conmigo de su amistad.

En fin agradecer a todos los que de una manera u otra contribuyeron al desarrollo de este trabajo y a los que confiaron siempre en mi para seguir adelante.

Dedicatoria

Dedico el logro de este trabajo a toda mi familia por estar siempre preocupados y atentos durante mi vida universitaria.

A mis padres por estar siempre a mi lado y darme fuerzas para seguir adelante en todo momento, por estar ahí cuando más necesitaba de ellos.

A mis abuelos y en especial a mima Sarah por ser mi otra madre y ser la principal impulsora de coger esta carrera, para ti es este resultado.

A mis amigos por brindarme todo el apoyo del mundo y por compartir los mejores momentos de mi vida.

Resumen

En el departamento de Control Automático del Instituto de Cibernética Matemática y Física (ICIMAF) se cuenta con un laboratorio docente que posee una variedad de prácticas de laboratorios para cada uno de los equipos existentes. Entre estos equipos se encuentra un motor de corriente directa, que es una máquina capaz de convertir la energía eléctrica en mecánica. Actualmente existe para este motor un sistema informático que realiza su supervisión y control, dicho sistema fue implementado para ejecutarse solamente en las diferentes versiones del sistema operativo Windows, además de no ser un sistema en tiempo real.

Un sistema de monitoreo y control para realizar prácticas de laboratorios con un motor de este tipo debe ser un sistema lo más sencillo posible y en tiempo real debido a los períodos de muestreo críticos que presentan los dispositivos de control. Para la implementación de la solución se decidió utilizar la Interfaz en tiempo real para aplicaciones en Linux (RTAI). La presente investigación tiene como objetivo diseñar un sistema que monitoree y controle en tiempo real un motor de corriente directa a través de una tarjeta de adquisición de datos. Dicho sistema debe permitir conectarse al motor mediante el puerto serie, leer y regular los valores de la velocidad y por ciento de ciclo. Graficar los valores leídos en tiempo real, salvar las muestras realizadas en un fichero y cargarlas desde un archivo existente. Además permite realizar una simulación a partir de valores entrados manualmente por el usuario.

Palabras clave: motor, real, tiempo, velocidad.

Índice de contenido

Índice de contenidos

Índice de contenidos

Índice de figuras

Índice de tablas

Introducción.

Las técnicas de control han evolucionado en la medida en que la producción en los procesos industriales fue cambiando y vinculándose a tecnologías cada vez más sofisticadas. Este avance ha sido fortalecido por el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), de manera que los procesos productivos puedan adaptarse a las nuevas exigencias de rendimiento, comunicación, costos, integración e incluso a los requerimientos de protección ambiental.

Entre las tecnologías que apoyan estos procesos se encuentran los sistemas de control que son considerados Sistemas en Tiempo Real (STR) debido a las restricciones de tiempo a las que están sujetas las respuestas que este debe dar. Un sistema en tiempo real realiza el procesamiento de información teniendo que responder a estímulos de entrada generados externamente en un período finito y específico, en ellos las respuestas correctas dependen no sólo de los resultados lógicos sino también del tiempo en que son entregadas. Un ejemplo de STR son los sistemas empotrados que se desarrollan sobre microprocesadores dedicados, que tienen como única tarea controlar el sistema (1).

En ocasiones se necesita que el microprocesador atienda otras tareas como pueden ser la gestión de periféricos o la interfaz hombre máquina y por tanto se debe utilizar un Sistema Operativo en Tiempo Real (SOTR). Los SOTR brindan la posibilidad de controlar el tiempo en que se deben ejecutar las tareas. Los desarrolladores que emplean esta variante pueden aprovechar las funcionalidades que proporcionan los SOTR, como: mecanismos de comunicación entre procesos, planificadores y despachadores de interrupciones. Para la implementación de un STR también existen sofisticados ambientes especializados como los Sistemas de Control Distribuido (Distributed Control System – DCS, por sus siglas en inglés) en los cuales los componentes hardware y/o software ubicados en computadores en red, se comunican y coordinan sus acciones intercambiando mensajes o los Sistemas de Supervisión, Control y Adquisición de Datos (Supervisory Control And Data Acquisition – SCADA, por sus siglas en inglés) que ejecutan tareas críticas como la recolección periódica, procesamiento y monitoreo de información, control remoto de dispositivos de campo y visualización de alarmas.

En la actualidad existen numerosos grupos de investigación en universidades y empresas líderes de todo el mundo que diseñan y desarrollan nuevos sistemas de control automáticos para aplicaciones

complejas. En Cuba existen instituciones que se dedican a la investigación y desarrollo de estos sistemas entre los que resalta el Instituto de Cibernética, Matemática y Física (ICIMAF), entidad subordinada al Ministerio de Ciencia, Tecnología y Medio Ambiente (CITMA), que tiene como objetivo facilitar los avances en el desarrollo de las investigaciones teóricas y aplicadas. Desde su fundación hasta la fecha es de reconocida presencia por los resultados obtenidos en las investigaciones, servicios científicos y tecnológicos a nivel nacional e internacional.

El ICIMAF tiene como principal misión gestionar y ejecutar proyectos de investigación, desarrollo e innovación en matemática, física y cibernética con personal motivado y de competencia reconocida. Así como realizar actividades de formación posgraduada, asesorías y servicios científicos y tecnológicos que brinden soluciones de alto valor agregado. Como objeto social se distingue por brindar servicios de capacitación y superación de postgrado, producir y comercializar publicaciones en cualquier soporte científico técnico relacionadas en las temáticas especializadas. La institución cuenta con varios departamentos, entre ellos el de Control Automático que se encarga de la automatización de los procesos industriales mediante sistemas de control automático incluyendo teorías y aplicaciones (2).

Este departamento consta de un laboratorio docente que posee una variedad de prácticas de laboratorios para cada uno de los equipos existentes. Entre estos equipos se encuentra un motor de corriente directa, que es una máquina capaz de convertir la energía eléctrica en mecánica. Un sistema de monitoreo y control para realizar prácticas de laboratorios con un motor de este tipo debe ser un sistema en tiempo real por tanto ha de brindar la posibilidad de tener control del tiempo en que se deben ejecutar las tareas. Actualmente las prácticas de laboratorios con el motor de corriente directa se llevan a cabo haciendo uso de un sistema realizado para ejecutarse solo en el sistema operativo Windows y no posee la característica de ser un sistema en tiempo real.

A partir de lo anteriormente expuesto, se plantea el siguiente **problema de investigación:** ¿cómo monitorear y controlar en tiempo real un motor de corriente directa?

Este problema se enmarca en el **objeto de estudio:** sistemas en tiempo real para el monitoreo y control de un motor.

Delimitándose como **campo de acción:** sistemas en tiempo real para el monitoreo y control de un motor de corriente directa.

Definiendo como **objetivo general:** desarrollar un sistema informático que permita supervisar y controlar en tiempo real un motor de corriente directa.

Para lograr el cumplimiento del objetivo planteado anteriormente, se definen como principales **tareas de la investigación:**

- ✓ Elaboración del marco teórico de la investigación a partir del estado del arte existente sobre el tema.
- ✓ Análisis y estudio de las características de un motor de corriente directa.
- ✓ Identificación de los requerimientos funcionales y no funcionales del sistema y del protocolo de comunicación con el motor de corriente directa.
- ✓ Selección de las metodologías, herramientas de software y lenguajes de programación a utilizarse para el desarrollo del sistema informático.
- ✓ Implementación y diseño del sistema informático.
- ✓ Validación de las propuestas realizadas.

Para apoyar el desarrollo de la investigación se emplean los siguientes métodos de investigación científica.

Métodos Teóricos:

- ✓ **Analítico – Sintético:** Permite analizar las teorías y documentos referentes al tema de la investigación, facilitando de esta forma la extracción de los elementos más importantes relacionados con la misma.

- ✓ **Histórico – Lógico:** Permite realizar un estudio de los trabajos anteriores relacionados con el tema a investigar, lo que todos conocen en el campo de la investigación como el estudio del estado del arte. Dando la posibilidad de conocer acerca de la existencia y características de sistemas de este tipo que hayan sido desarrollados anteriormente en el mundo.

Métodos Empíricos:

Los métodos empíricos generales se utilizan a través de la observación y los particulares a través de las entrevistas como guía de orientación. Se aplica el análisis documental y bibliográfico para la asimilación y estudio de la documentación del proyecto.

El presente trabajo de diploma se divide en tres capítulos, los cuales estarán estructurados de la siguiente forma:

Capítulo 1. Fundamentación teórica: en este capítulo se describen los principales conceptos a tratar y se aborda el estado del arte de los motores de corriente directa, y de los sistemas en tiempo real existentes en el ámbito nacional e internacional. Además se define la metodología de desarrollo, herramientas y tecnologías a utilizar para la realización del sistema.

Capítulo 2. Características, análisis y diseño del sistema: en este capítulo se exponen las características funcionales del sistema mediante las Historias de Usuario (HU) así como las no funcionales. Se realiza una propuesta del prototipo no funcional de la aplicación, y se estiman y planifican las HU para su posterior implementación, definiendo la arquitectura del sistema y los patrones de diseño a emplear.

Capítulo 3. Implementación y prueba: en este capítulo se organizan las clases fundamentales del sistema. Se procede a desarrollar las tareas de la ingeniería que corresponden a las HU desarrolladas anteriormente, luego mediante las pruebas se verifica que el producto resultante cumpla con los requisitos definidos.

Capítulo 1: Fundamentación teórica.

1.1. Introducción.

En el presente capítulo se realiza un estudio profundo de las actuales soluciones informáticas a nivel nacional e internacional que están relacionadas con el monitoreo y control de sistemas en tiempo real. Se fundamenta acerca de aspectos y estilos arquitectónicos, tecnologías, herramientas, metodologías y lenguajes que son empleados en el desarrollo de la solución. Además de analizar y discutir los conceptos fundamentales relacionados con el tema a investigar, para proponer una definición propia de los mismos.

1.2. Conceptos asociados a la investigación.

Los sistemas de monitoreo y control abarcan numerosos conceptos que son imprescindibles y fundamentales para el desarrollo de este tipo de aplicaciones. A continuación se muestran algunos de los más importantes.

1.2.1. Monitoreo y control.

El monitoreo consiste en la observación de uno o más parámetros para detectar eventuales anomalías (3). Mientras que el control es un mecanismo que detecta cualquier desvío de los patrones normales, haciendo posible la debida regulación (4).

Actualmente existen sistemas para el monitoreo y control de procesos o equipos que permiten determinar y analizar de manera eficiente y rápida las causas que pueden originar un mal funcionamiento o errores que estos puedan presentar. Para el seguimiento y evaluación del comportamiento de estos equipos que se desean monitorear y controlar se pueden definir variables, indicadores, estándares, tiempo o intervalos de monitoreo, gráficos de resultados, entre otros componentes.

Estas acciones de monitorear y controlar un sistema o equipo permite la detección de problemas reduciendo las posibilidades de fracaso, posibilitando la adopción oportuna de medidas correctivas reduciendo la probabilidad de fallas catastróficas.

En fin, el monitoreo y control se basa en la supervisión y regulación de parámetros de un proceso o equipo para mantener estables sus condiciones previendo dificultades o fallos que puedan ocurrir.

1.2.2. Sistemas en Tiempo Real.

Los STR, tienen la capacidad de interactuar activamente con un entorno de dinámica conocida en relación con sus entradas, salidas y restricciones temporales para darle un correcto funcionamiento de acuerdo con los conceptos de predicción, estabilidad, control y alcance. Son un elemento imprescindible para asegurar la calidad y la seguridad de incontables procesos industriales (5).

La principal característica que distingue a los STR de otros tipos de sistemas es el tiempo de interacción. La palabra "tiempo" significa que el correcto funcionamiento de un sistema depende no solo del resultado lógico que devuelve la computadora, también depende del tiempo en que se produce ese resultado. La palabra "real" quiere decir que la reacción de un sistema a eventos externos debe ocurrir durante su evolución. Como una consecuencia, el tiempo del sistema (tiempo interno) debe ser medido usando la misma escala con que se mide el tiempo del ambiente controlado (tiempo externo).

Se puede resumir que los STR son aquellos sistemas informáticos que controlan un ambiente que tiene restricciones de tiempo bien definidas, es por ello, que se vuelven más complejos y por lo tanto demandan una alta confiabilidad, con resultados correctos, predecibles y a tiempo.

1.2.3. Sistemas Operativos en Tiempo Real (SOTR).

Un sistema operativo de tiempo real es un sistema operativo que ha sido desarrollado para aplicaciones de tiempo real. Como tal, se le exige corrección en sus respuestas bajo ciertas restricciones de tiempo. Para garantizar el comportamiento correcto en el tiempo requerido se necesita que el sistema sea predecible (6).

Los sistemas operativos de tiempo real brindan la posibilidad de tener control del tiempo en que se deben ejecutar las tareas. En el mercado actual existe una gran variedad de sistemas operativos de tiempo real. Entre ellos, se destacan: RTLinux, ADEOS, QNX (en todos los casos software propietario), y RTAI (software libre) (7).

Teniendo en cuenta los conceptos antes descritos, los SOTR son sistemas informáticos que tienen la capacidad de interactuar rápidamente con su entorno físico, cumpliendo con las restricciones de tiempo establecidas.

1.2.4. Interfaz en tiempo real para aplicaciones en Linux. RTAI 3.8.1

RTAI (por sus siglas en inglés Real Time Application Interfaces) es una implementación de Linux para tiempo real basada en RTLinux que añade un pequeño kernel de tiempo real bajo el kernel estándar de Linux y trata al kernel de Linux como una tarea con la menor prioridad (Figura 1). RTAI además proporciona una amplia selección de mecanismos de comunicación entre procesos y otros servicios de tiempo real. Adicionalmente proporciona un módulo llamado LXRT[1] para facilitar el desarrollo de aplicaciones de tiempo real en el espacio de usuario (8).

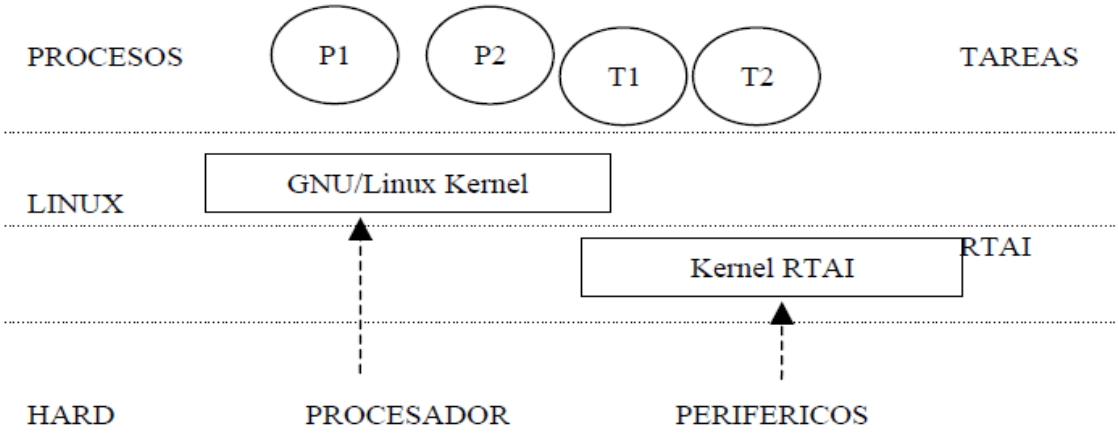


Figura 1. Arquitectura de un sistema operativo GNU/Linux con un parche RTAI.

RTAI tiene una arquitectura similar a RTLinux, trata el kernel estándar de Linux como una tarea de tiempo real con la menor prioridad, lo que hace posible que se ejecute cuando no haya ninguna tarea con mayor prioridad ejecutándose. Las operaciones básicas de las tareas de tiempo real son implementadas como módulos del kernel al igual que RTLinux. RTAI maneja las interrupciones de periféricos y son atendidas por el kernel de Linux después de las posibles acciones de tiempo real que hayan podido ser lanzadas por efecto de la interrupción.

Características de RTAI.

Los diferentes mecanismos de IPC (por sus siglas en inglés Inter-Process Comunication) están incluidos como módulos de núcleo, lo que facilita la carga cuando son necesarios. Como ventaja adicional, el uso de módulos para los servicios de comunicación entre procesos IPC facilita el mantenimiento y la expansión.

[1] **LXRT**: API para RTAI que hace posible el desarrollo de aplicaciones de tiempo real en el espacio de usuario sin tener que crear módulos para el núcleo.

El método más flexible de IPC son los buzones. Estos procesos pueden enviar y recibir mensaje de y desde un buzón. Un buzón almacena mensajes hasta un límite que se defina, puede haber un número arbitrario de buzón activos en el sistema simultáneamente. RTAI también facilita la comunicación entre procesos mediante RPC (Por sus siglas en inglés Remote Procedure Call).

Las actuales versiones de RTAI incluyen un módulo gestor de memoria que permite la asignación dinámica de memoria por parte de las tareas de tiempo real usando un interfaz basado en una biblioteca estándar de C (9).

Aplicaciones de RTAI.

Son aplicadas en especial en el área del control, robótica y para sistemas militares, en sistemas que requieren toma de datos y procesamiento de estos en tiempo real.

Se utilizan en todos los sistemas de control, por ejemplo, control de plantas, motores, y para el monitoreo de todo tipo de sistemas, ya sean médicos, de medición, u otros. En la robótica donde se debe tomar mediciones del medio, procesar la información, y con tal información tomar decisiones en tiempo real. También es bastante utilizado en sistemas militares donde se requiere gran capacidad de cómputo para tomar decisiones en tiempo real ante la toma de datos del exterior. Otra área en la que se utiliza con gran amplitud RTAI es en los sensores de red.

Con el gran crecimiento de los sistemas integrados y del control digital, podemos decir que en este momento casi todos los sistemas electrónicos de última tecnología cuentan u operan con este sistema, ya que gracias a su flexibilidad permite su implementación en varios procesadores. En la actualidad ya existen varios microprocesadores que soportan este tipo de sistemas, permitiendo un gran desarrollo de sistemas electrónicos en general.

1.2.5. Sistemas Empotrados o Embebidos.

Los sistemas empotrados o embebidos son una combinación de hardware y software de computador, también llamado plataforma de cómputo, sumado generalmente a algunas piezas mecánicas o sistema de ingeniería más amplio. Los sistemas empotrados están diseñados para tener una función específica o dedicada. Esta combinación de software y hardware puede ser reemplazada en muchos casos por un circuito integrado que cumpla la misma tarea. Pero una de las ventajas de los sistemas embebidos es su flexibilidad, debido que a la hora de realizar alguna modificación es mucho más fácil modificar algunas líneas de código al software del sistema embebido que reemplazar todo un circuito integrado (10).

Características más importantes:

- ✓ Concurrencia: todos los componentes del sistema monitoreado recurren al sistema embebido al mismo tiempo y este último debe actuar en consecuencia.
- ✓ Eficacia: deben responder con gran rapidez a los cambios en el sistema controlado.
- ✓ Bajo Consumo: estos sistemas generalmente son de bajo consumo (mayor autonomía).
- ✓ Precio Bajo: son de precios relativamente bajos dados sus funcionalidades, esto varía según la empresa distribuidora.
- ✓ Tamaño Pequeño: Los sistemas empotrados tienen muy pocos recursos de memoria y E/S. El sistema empotrado debe ajustarse al sistema que monitorea (11).

Áreas en las que son utilizados los sistemas embebidos:

- ✓ Electrónica de consumo: Videos, HIFI, televisión, lavadoras, frigoríficos, lavaplatos.
- ✓ Automóviles: Control velocidad, climatización, visualización, inyección.
- ✓ Telecomunicaciones: Radios, teléfonos móviles, GPS (por sus siglas en inglés Global Positioning System).
- ✓ Aeronáutica: Computadores de vuelo y de misión.
- ✓ Defensa: Bombas y misiles inteligentes, vehículos con dirección de tiro (12).

Se puede concluir que los sistemas embebidos han tenido un gran avance tecnológico para la humanidad ya que han liberado de muchas tareas complejas que el hombre generalmente no puede o se le hace muy complicado efectuar por ser muy complejas, mientras que un sistema embebido lo cumple fácilmente.

1.2.6. Motor de corriente directa.

Un motor eléctrico es una máquina que transforma energía eléctrica en energía mecánica, que generalmente es un movimiento rotatorio, por medio de interacciones electromagnéticas. Algunos de los motores eléctricos son reversibles, es decir, pueden transformar energía mecánica en energía eléctrica funcionando como generadores (13). Pueden funcionar conectados a una red de suministro eléctrico o con baterías. Son ampliamente utilizados en instalaciones industriales, comerciales y particulares para

satisfacer una amplia gama de necesidades de servicio, desde arrancar, acelerar, mover o frenar, sostener y detener una carga.

Los motores se pueden clasificar según el tipo de alimentación que tengan, de esta forma, se tienen los motores de corriente continua y los de corriente alterna. Un motor de corriente directa (CD), o motor de corriente continua (CC), es un dispositivo que transforma la energía eléctrica en mecánica por medio de interacciones electromagnéticas. Entre los motores (CD), está el derivado, el de serie, el compuesto y el de imán permanente.

En este trabajo, se utiliza un motor de corriente continua que trabaja como maqueta docente, funcionando con una aplicación informática que permite como principal objetivo, controlar su velocidad angular.

Lo fundamental de un motor de corriente continua son sus dos polos, entre los cuales existe un campo magnético formado por los dos de un imán (positivo y negativo). Esta fuerza es aprovechada por un motor eléctrico haciendo girar un eje, cuyo proceso conlleva a la transformación de la energía eléctrica en movimiento mecánico.

Partes de un motor de CC.

Los motores de corriente continua están constituidos por dos elementos básicos de cualquier motor eléctrico que son el rotor (eje interno) y el estator (imán externo o bobina). La principal característica del motor de corriente continua es la posibilidad de regular la velocidad desde vacío a plena carga (13).

A continuación se muestra una imagen que indica las principales partes de un motor de este tipo.

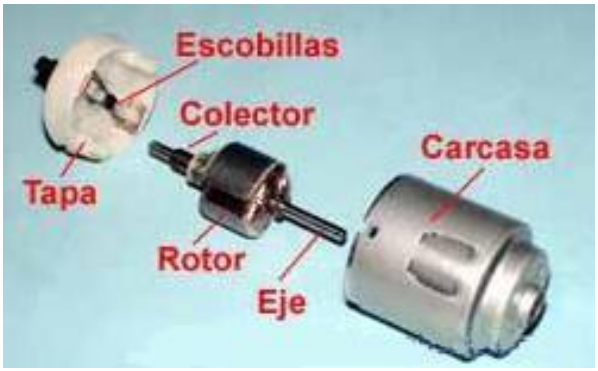


Figura 2. Partes principales de un motor de CC.

A continuación se ofrece una breve explicación de cada una de estas partes (15).

Partes del Rotor.

Figura 3. Rotor.

Rotor: es la parte móvil del motor, proporciona el torque para mover a la carga. Es generalmente de forma cilíndrica, también devanado y con núcleo, al que llega la corriente mediante dos escobillas. Eje: formado por una barra de acero fresada. Imparte la rotación al núcleo, devanado y colector.

Núcleo: Se localiza sobre el eje. Hecho con capas laminadas de acero, cuya función es proporcionar un trayecto magnético entre los polos para que el flujo magnético del devanado circule.

Devanado: Consta de bobinas aisladas entre sí y entre el núcleo de la armadura. Estas bobinas están alojadas en las ranuras y están conectadas eléctricamente con el colector, el cual, debido a su movimiento rotatorio, proporciona un camino de conducción conmutado.

El Colector: Denominado también conmutador, está constituido de láminas de material conductor (delgas), separadas entre sí y del centro del eje por un material aislante, para evitar cortocircuito con dichos elementos. El colector se encuentra sobre uno de los extremos del eje del rotor, de modo que gira con este y está en contacto con las escobillas. La función del colector es transmitir la tensión producida por el devanado inducido, al circuito (carga del motor) por medio de las escobillas (llamadas también cepillos).

Partes del Estator.

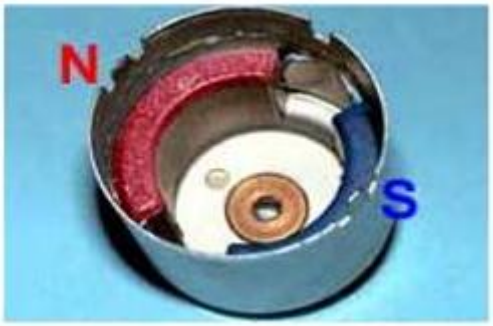


Figura 4. Estator.

Estator: Constituye la parte fija de la máquina. Su función es suministrar el flujo magnético que será usado por el bobinado del rotor para realizar su movimiento giratorio.

Armazón: Denominado también carcasa o yugo, tiene dos funciones primordiales: servir como soporte y proporcionar una trayectoria de retorno al flujo magnético del rotor y del imán permanente, para completar el circuito magnético.

Imán permanente: Compuesto de material ferromagnético altamente remanente, se encuentra fijado al armazón o carcasa del estator. Su función es proporcionar un campo magnético uniforme al devanado del rotor o armadura, de modo que interactúe con el campo formado por el bobinado, y se origine el movimiento del rotor como resultado de la interacción de estos campos.

Escobillas: Las escobillas están fabricadas se carbón, y poseen una dureza menor que la del colector, para evitar que éste se desgaste rápidamente. Se encuentran albergadas por las porta escobillas. Ambas se encuentran en una de las tapas del estator. La función de las escobillas es transmitir la tensión y corriente de la fuente de alimentación hacia el colector y, por consiguiente, al bobinado del rotor. La función de las porta escobillas es mantener a las escobillas en su posición de contacto firme con los segmentos del colector.

Clasificación de los motores de CC.

Los motores convencionales de corriente continua, pueden clasificarse sobre la base de las conexiones eléctricas mutuas entre los devanados de campo y armadura. El devanado o circuito de campo es aquel que proporciona un flujo magnético para que el rotor pueda girar constantemente. El circuito de armadura es donde se encuentra la alimentación del motor y proporciona el primer arranque.

La siguiente imagen muestra un esquema con la clasificación de los motores de CC.

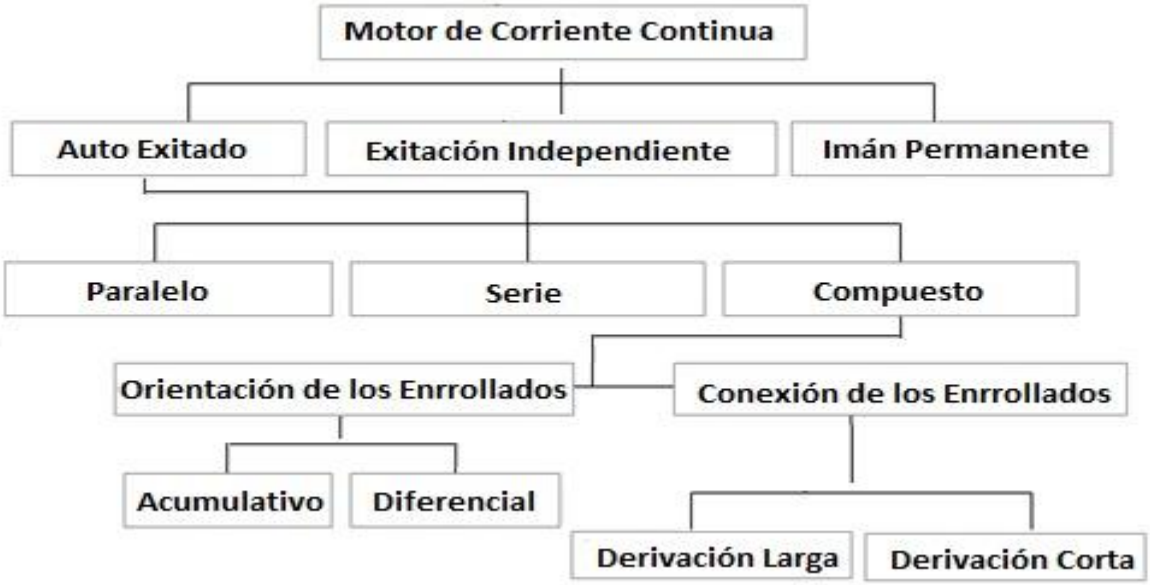


Figura 5. Clasificación de los motores de Corriente Continua (CC).

1.3. Estado del arte de los sistemas de monitoreo y control.

La Ingeniería en Automatización y Control Industrial es una carrera que cada día se ve con mayor demanda en el ámbito industrial debido a que los procesos de producción que tienen las empresas están en una constante carrera contra el tiempo debido a que los retardos en los procesos de producción en algunas empresas pueden incluso generar grandes pérdidas de carácter monetario.

Entre las áreas donde se desarrolla esta disciplina se destacan sectores industriales en rubros[2] como la Minería, Celulosa, Metalmecánica, Automotriz, Textil, Alimentos, Integración Ingenieril entre otras que requieran de una optimización en su sistema de producción (14).

Con el gran avance tecnológico en el mundo y el elevado desarrollo industrial, se han provocado numerosos cambios en los sectores empresariales de gran escala mundial. El monitoreo y control es una pieza fundamental para el desarrollo de muchas empresas, ya que muchas de estas dependen de sistemas capaces que puedan ser monitoreados y controlados para su funcionamiento.

[2] **Rubro:** categoría que permite reunir en un mismo conjunto a entidades que comparten ciertas características.

1.3.1. Ámbito internacional.

Mundialmente existen varios sistemas de monitoreo y control para múltiples ramas, siendo estos sistemas una herramienta potente para la predicción del mantenimiento de los equipos en el momento justo que se requiera.

Sistema central de monitoreo y control remoto para equipos de aire acondicionado.
Este sistema ha sido desarrollado en México para monitorear y controlar en tiempo real estos equipos, entre las condiciones de operación que monitorea está las temperaturas del espacio acondicionado, el consumo de energía, entre otros parámetros. También posibilita llevar a cabo acciones de control, como el encendido y apagado del equipo. Asimismo se puede evaluar el desempeño del equipo y detectar condiciones anormales de operación que requieren acción inmediata.

Sistema de riego automatizado en tiempo real.
En el Colegio de Postgraduados del Campus Montecillo, México, se desarrolló un sistema de riego automatizado en tiempo real para determinar, controlar el momento oportuno y la cantidad de riego, monitoreado por medio de las tecnologías de información. Se monitorea mediante tres estrategias implementadas: balance hídrico, medición de humedad del suelo y lisímetro. Gracias al monitoreo continuo del sistema de riego y de los puertos de control en tiempo real se logró dar un seguimiento puntual al estado de los cultivos.

Sistemas SCADA reconocidos a nivel mundial
SIMATIC WinCC de la compañía Siemens, GENESIS 32 de IONICS, InTouch de Wonderware. Todos los sistemas mencionados anteriormente son software privativo y únicamente se ejecutan sobre Windows.

1.3.2. Ámbito nacional.

Sistema de monitoreo y control en tiempo real para un horno.
Este sistema es un controlador lógico programable desarrollado en software libre, y con la característica similar de trabajar en tiempo real. Permite visualizar los procesos de control, los parámetros internos son la humedad y la temperatura del horno. Si estos parámetros están fuera de rango no se puede pasar al control interno del horno. La interfaz del sistema incluye no solo el monitoreo sino también el control, pues es aquí donde se ejecutan los ensayos con el horno.

Sistema Movicorde

Movicorde es un sistema para el monitoreo electrocardiográfico por telemetría, es una herramienta de ayuda a los médicos rehabilitadores, cardiólogos, fisioterapeutas, y personal de enfermería especializado durante el proceso de rehabilitación de los enfermos coronarios en las unidades de rehabilitación cardíaca. Eventualmente puede ser empleado para el control de la preparación técnica de atletas en centros de entrenamiento. La utilización de la telemetría para la supervisión de los ejercicios físicos controlados en una unidad de rehabilitación cardíaca permite a los especialistas monitorear el electrocardiograma en tiempo real, ofreciéndoles la posibilidad de observar la evolución de los pacientes y elaborar diagnósticos avanzados que les informen acerca de la presencia de factores de riesgo cardiovascular.

Sistema SCADA "DADIVA"

El Centro de Investigación y Desarrollo Naval (CIDNAV) con vistas al impacto que han creado las nuevas tecnologías en la Ingeniería Naval y continuando con el avance de las mismas, creó un SCADA dirigido especialmente a los sistemas ingenieros navales de los buques de la Marina cubana. Los objetivos principales del sistema son:

- ✓ Analizar en tiempo real en funcionamiento de la Planta.
- ✓ Detectar anomalías y/o fallas en la misma.
- ✓ Realizar estudios profilácticos con el fin de realizar estadísticas de fallas y determinar acciones de mantenimiento a las instalaciones energéticas de los buques.

Sistema SCADA "GALBA"

El GALBA es un sistema SCADA desarrollado en la Universidad de las Ciencias Informáticas (UCI) para la Empresa Socialista Mixta de Venezuela. Este sistema es responsable de la supervisión y control del proceso industrial de la Empresa Petróleos de Venezuela S.A. (PDVSA). El GALBA es un sistema de tiempo real, distribuido en módulos que trabajan de manera conjunta posibilitando el funcionamiento del sistema como un todo. Estos módulos se encuentran interconectados a través de un software para la distribución de los servicios en la red, conocido como "middleware" o software de comunicación entre aplicaciones.

En nuestro país existen implementaciones de sistemas SCADA de mediana complejidad como "Titán" creado por el grupo Inel de Villa Clara y "Eros" desarrollado por el grupo Eros de Nicaro Holguín. Ambos sistemas corren sobre Windows.

1.3.3. Análisis sobre los sistemas de monitoreo y control.

Después de haber realizado un estudio profundo sobre los sistemas de monitoreo y control se concluye que la mayoría de estos sistemas corren sobre plataforma Windows. Aunque existen algunas implementaciones para Linux como es el caso particular del SCADA “GALBA” desarrollado en la UCI no se considera factible su utilización debido a que para su utilización se debe implementar un driver que permita la comunicación por puerto serie con el motor. En caso de que se implementara dicho driver se tendrían que instalar y configurar varios módulos para el correcto funcionamiento del mismo lo que haría muy compleja la solución. A pesar de que este sistema se puede instalar gratis, no se cuenta con el código fuente debido a que es propiedad de la Empresa Socialista Mixta de Venezuela, esto impide ampliar dicho sistema mediante la agregación de nuevas funcionalidades. Por todo lo antes mencionado se decide desarrollar el sistema de monitoreo y control en tiempo real usando la Interfaz en tiempo real para aplicaciones en Linux RTAI que garantiza que el sistema pueda ejecutarse en la plataforma de Linux, además de correr en tiempo real.

1.4. Plataformas de desarrollo del software.

El Proyecto Debian es una asociación de personas con el objetivo de crear sistemas operativos (SO) libres. Un sistema operativo es un conjunto de programas y utilidades básicas que hacen que una computadora funcione. El centro de un sistema operativo es el núcleo, este es el programa más importante en la computadora, ya que realiza todo el trabajo básico y le permite ejecutar otros programas. Los sistemas Debian actualmente usan el núcleo de Linux o de FreeBSD. Linux es un sistema operativo creado por Linux Torvalds y desarrollada por miles de programadores a lo largo del mundo. FreeBSD es un sistema operativo que incluye un núcleo y otro software. Una gran parte de las herramientas básicas que completan el sistema operativo, vienen del proyecto GNU; de ahí los nombres: GNU/Linux, GNU/kFreeBSD y GNU/Hurd (15).

Con la investigación realizada y con la deducción de utilizar un sistema operativo Linux sobre el cual se pudiera parchear RTAI, se hizo uso de **Debian 6.0** como sistema operativo para la realización del software, ya que es uno de los sistemas operativos con mayores perspectivas y con más estabilidad de la familia de los sistemas operativos libres.

1.5. Descripción del lenguaje de programación.

Los lenguajes de programación son la herramienta informática que nos permite establecer una correcta comunicación entre la computadora y el usuario de la misma. Actualmente existen numerosos lenguajes de programación, entre los que se pueden mencionar:

C#:

Fue creado por el danés Anders Hejlsberg que diseño también los lenguajes Turbo Pascal y Delphi. El C# (pronunciado en inglés “C sharp” o en español “C sostenido”) es un lenguaje de programación orientado a objetos. Con este lenguaje se quiso mejorar con respecto de los dos lenguajes anteriores de los que deriva el C y el C++. Se consiguió que tuviese las ventajas del C y del C++.

Características del lenguaje de programación C#:

- ✓ Su código se puede tratar íntegramente como un objeto.
- ✓ Su sintaxis es muy similar a la del JAVA.
- ✓ Es un lenguaje orientado a objetos y a componentes.
- ✓ Armoniza la productividad del Visual Basic con el poder y la flexibilidad del C++.
- ✓ Ahorro de tiempo en la programación ya que tiene una librería de clases muy completa y bien diseñada.
- ✓ Forma parte de la plataforma .NET, que es una interfaz de programación de aplicaciones. C# es un lenguaje independiente que originariamente se creó para producir programas sobre dicha plataforma.

Java:

El lenguaje Java tiene su propia estructura, reglas de sintaxis y paradigma de programación. El paradigma de programación del lenguaje Java se basa en el concepto de programación orientada a objetos (OOP), que las funciones del lenguaje soportan.

El lenguaje Java es un derivado del lenguaje C, por lo que sus reglas de sintaxis se parecen mucho a C: por ejemplo, los bloques de códigos se modularizan en métodos y se delimitan con llaves ({ y }) y las variables se declaran antes de que se usen.

Estructuralmente, el lenguaje Java comienza con paquetes. Un paquete es el mecanismo de espacio de nombres del lenguaje Java. Dentro de los paquetes se encuentran las clases y dentro de las clases se encuentran métodos, variables, constantes, entre otros. En este tutorial, aprenderá acerca de las partes del lenguaje Java.

Características del lenguaje Java:

- ✓ La principal característica de Java es la de ser un lenguaje compilado e interpretado. Todo programa en Java ha de compilarse y el código que se genera bytecodes es interpretado por una máquina virtual. De este modo se consigue la independencia de la máquina, el código compilado se ejecuta en máquinas virtuales que si son dependientes de la plataforma.

- ✓ Java es un lenguaje orientado a objetos de propósito general. Aunque Java comenzará a ser conocido como un lenguaje de programación de applets que se ejecutan en el entorno de un navegador web, se puede utilizar para construir cualquier tipo de proyecto.

- ✓ Su sintaxis es muy parecida a la de C y C++ pero hasta ahí llega el parecido. Java no es una evolución ni de C++ ni un C++ mejorado.

- ✓ Todas las instancias de una clase se crean con el operador new(), de manera que un recolector de basura se encarga de liberar la memoria ocupada por los objetos que ya no están referenciados. La máquina virtual de Java gestiona la memoria dinámicamente.

- ✓ Una fuente común de errores en programación proviene del uso de punteros. En Java se han eliminado los punteros, el acceso a las instancias de clase se hace a través de referencias. Además, el programador siempre está obligado a tratar las posibles excepciones que se produzcan en tiempo de ejecución. Java define procedimientos para tratar estos errores.

- ✓ Posee mecanismos para garantizar la seguridad durante la ejecución, comprobando antes de ejecutar el código, que este no viole ninguna restricción de seguridad del sistema donde se va a ejecutar.

- ✓ Cuenta con un cargador de clases, de modo que todas las clases cargadas a través de la red tienen su propio espacio de nombres para no interferir con las clases locales.

- ✓ Otra característica de Java es que está preparado para la programación concurrente sin necesidad de utilizar ningún tipo de biblioteca.

- ✓ Finalmente, Java posee un gestor de seguridad con el que poder restringir el acceso a los recursos del sistema.

C++:

Es una extensión del lenguaje de programación C. Las principales características del C++ son el soporte para programación orientada a objetos y el soporte de plantillas o programación genérica. Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

Es un lenguaje versátil y potente. Su éxito entre los programadores profesionales le ha llevado a ocupar el primer puesto como herramienta de desarrollo de aplicaciones. El C++ mantiene las ventajas del C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Además, ha eliminado algunas de las dificultades y limitaciones del C original.

Para la implementación del sistema se tomó como lenguaje de programación C++, ya que de los tres lenguajes analizados es el más rápido en cuanto a tiempo de ejecución pues los otros son lenguajes interpretados, lo que hace que su ejecución sea más lenta.

1.6. Metodología de desarrollo.

Las metodologías ágiles de desarrollo están especialmente indicadas en proyectos con requisitos poco definidos o cambiantes. Estas metodologías se aplican bien en equipos pequeños que resuelven problemas concretos, lo que no está reñido con su aplicación en el desarrollo de grandes sistemas, ya que una correcta modularización de los mismos es fundamental para su exitosa implantación. Dividir el trabajo en módulos abordables minimiza los fallos y el coste. Las metodologías ágiles presentan diversas ventajas, entre las que se destacan:

- ✓ Capacidad de respuesta a cambios de requisitos a lo largo del desarrollo.
- ✓ Entrega continua y en plazos breves de software funcional.
- ✓ Trabajo conjunto entre el cliente y el equipo de desarrollo.
- ✓ Importancia de la simplicidad, eliminado el trabajo innecesario.
- ✓ Mejora continua de los procesos y el equipo de desarrollo.

La siguiente imagen representa una tabla que recoge esquemáticamente las principales diferencias de las metodologías ágiles con respecto a las tradicionales (no ágiles). Estas diferencias que afectan no sólo al proceso en sí, sino también al contexto del equipo así como a su organización (16).

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Figura 6. Diferencias entre metodologías ágiles y no ágiles.

Proceso Unificado de Desarrollo (RUP): Es un proceso para el desarrollo de un proyecto de software que define claramente quien, cómo, cuándo y qué debe hacerse en el proyecto. Como característica esencial este se dirige por los casos de uso; que orientan el proyecto a la importancia para el usuario y lo que este quiere, está centrado en la arquitectura; que relaciona la toma de decisiones que indican, cómo tiene que ser construido el sistema y en qué orden, es iterativo e incremental; dividiendo el proyecto en mini proyectos donde los casos de uso y la arquitectura cumplen sus objetivos de una manera más depurada (17).

XP (Extreme Programming): Una de las metodologías ágiles de desarrollo de software más exitosa en la actualidad y ampliamente utilizada a nivel mundial es Programación Extrema (XP), la cual se basa en la simplicidad, la comunicación y reutilización del código desarrollado. Evita elevados e innecesarios costos ya que muchas veces en otras metodologías se emplea demasiado tiempo y recursos en cambiar la documentación de la planificación para que se asemeje al código; para esto, intenta implementar una forma de trabajo que se adapte fácilmente a las circunstancias aumentando considerablemente la productividad del equipo.

El proceso de desarrollo XP brinda las siguientes ventajas:

- ✓ No presenta resistencia a cambios durante el proceso de desarrollo.
- ✓ Presenta iteraciones cortas que permiten obtener versiones funcionales del producto.
- ✓ Planificación más transparente para los clientes, conocen las fechas de entrega de funcionalidades vitales para su negocio.
- ✓ La presión está a lo largo de todo el proyecto y no en una entrega final (18).

El equipo de desarrollo para la presente investigación está compuesto por un desarrollador, el cliente para el cual se conformará la aplicación (Departamento de Control Automático) se encuentra vinculado fuertemente con el equipo de desarrollo. Además de que la disponibilidad de tiempo es limitada debido a que se necesita una aplicación sencilla funcional en el menor tiempo posible.

Basándose en lo anteriormente expuesto y teniendo en cuenta que se trata de un proyecto pequeño, donde la comunicación y el intercambio de opiniones con el cliente juegan un papel fundamental a lo largo de todo el proceso de desarrollo de software, con el propósito primordial de adquirir un producto con la calidad requerida, satisfaciendo las necesidades del cliente en el menor tiempo posible, se decide que la metodología XP guie todo el proceso de desarrollo del software.

1.7. Entorno de Desarrollo Integrado (IDE).

El Entorno de Desarrollo Integrado (IDE por sus siglas en inglés Integrated Development Environment), es un programa compuesto por un conjunto de herramientas de programación útiles para el desarrollador de software, con el objetivo de mejorar la productividad del desarrollador y obtener mayor rapidez en el desarrollo ofreciéndole un conjunto de herramientas totalmente cohesionadas entre sí, a través de un interfaz gráfico de usuario.

Existen algunos entornos compatibles con múltiples lenguajes de programación, como Visual Studio, Eclipse o NetBeans, ambos basados en Java; o MonoDevelop, basado en C#. También puede incorporarse la funcionalidad para lenguajes alternativos mediante el uso de plugins. Por ejemplo, Eclipse y NetBeans tienen plugins para C, C++, Ada, Perl, Python, Ruby y PHP entre otros; o Visual Studio que soporta no solo múltiples lenguajes si no también múltiples dispositivos.

Para la implementación del software se tenía que tener en cuenta que el IDE cumpliera con la política de software libre, pudiera usarse en Debian y RTAI. Para la adquisición de los datos era necesario que el entorno permitiera la comunicación mediante el puerto serie.

Qt Creator 2.5.0. Fue el IDE seleccionado porque permite que un equipo de desarrolladores comparta un proyecto a través de diferentes plataformas de desarrollo, admite la colaboración entre diseñadores y programadores. Cuenta con numerosas herramientas de avance de software. Qt constituye un framework de desarrollo multiplataforma usado para desarrollar aplicaciones con interfaz gráfica de usuario, así como también para el desarrollo de programas sin interfaz gráfica, como herramientas para la línea de comandos, consolas para servidores, entre muchos otros (19). El lenguaje que utiliza es C++, un lenguaje muy potente y completo.

1.8. Herramienta CASE para el modelado.

Las herramientas CASE[3] actualmente brindan una gran cantidad de componentes que incluyen la mayoría de los requisitos necesarios para el desarrollo de los sistemas, han sido creadas con una gran exactitud en cuanto a las necesidades de los desarrolladores de software para la automatización de procesos incluyendo el análisis, diseño e implantación. En la actualidad muchas empresas se han extendido a la adquisición de herramientas CASE, con el fin de automatizar sus procedimientos administrativos. Dentro de estas herramientas se encuentra Visual Paradigm for UML.

Visual Paradigm for UML

Se seleccionó **Visual Paradigm 8.0** debido a que es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite diseñar todos los tipos de diagramas (diagramas de casos de uso, diagramas de requerimiento y diseño de bases de datos relacionales), código inverso, generar código desde diagramas y generar documentación. Es una herramienta multiplataforma, muy fácil de usar y con un ambiente gráfico agradable para el usuario. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste (20).

1.9. Consideraciones finales.

Se realizó un estudio de las tendencias actuales de los sistemas en tiempo real y de las aplicaciones existentes para el monitoreo y control, ello posibilitó obtener elementos de referencia útiles para el

[3] **CASE:** siglas en inglés que se utilizan para referirse a Ingeniería de Software Asistida por Computadora.

diseño de la aplicación a desarrollar. Se analizaron las características de las tecnologías y metodologías actuales disponibles para el desarrollo de aplicaciones de este tipo, seleccionándose para ser utilizadas en este trabajo: el IDE QT Creator, el lenguaje de implementación C++, el lenguaje de modelado UML y la herramienta CASE Visual Paradigm, que se perfilaron como las más adecuadas para construir el sistema, dadas las características del mismo.

Capítulo 2: Características, análisis y diseño del sistema.

2.1. Introducción.

En el presente capítulo se abordan los temas relacionados con la fase de planificación de la metodología de desarrollo XP, se elabora una propuesta del sistema a desarrollar y se exponen las características del mismo para un mejor entendimiento en su desarrollo.

Se confeccionan las Historias de Usuarios (HU) para cada iteración definida, con vista a documentar los procedimientos y técnicas empleadas, proporcionando una mejor visión sobre lo que el cliente desea y además se realiza un análisis de la estimación del esfuerzo por cada HU.

2.2. Propuesta de solución.

Se propone implementar un sistema informático que permita supervisar y monitorear el comportamiento en tiempo real de un motor de corriente directa, y que permita además visualizar los resultados obtenidos una vez en marcha la aplicación.

2.3. Personal relacionado con el sistema.

La aplicación va dirigida a los estudiantes de Control Automático que cursan la maestría en Cibernética Aplicada “Mención Control Avanzado” del Instituto de Cibernética, Matemática y Física (ICIMAF), los cuales podrán interactuar con la misma en las prácticas que se realicen en el laboratorio.

2.4. Entorno de despliegue de la solución desarrollada.

El proceso objeto de estudio es un motor de corriente continua (CC) de imanes permanentes. Este motor no se utiliza en ningún proceso industrial, sino que funciona como maqueta de estudio en el Instituto de Cibernética Matemática y Física (ICIMAF) de La Habana, Cuba.
En el Anexo IV se muestran fotos del motor real que utilizará la solución desarrollada como parte de este trabajo de diploma.

La computadora (PC), donde se encontrará instalada la aplicación informática se encargará de controlar al motor, la comunicación será vía RS-232 con la tarjeta de adquisición de datos (TAD), compuesta, entre otros circuitos, por un microcontrolador PIC (por sus siglas en inglés Peripheral Interface

Controller), un circuito de potencia y un acondicionador de señal (AS). La Figura 7 muestra un esquema más detallado de los elementos que componen la TAD.

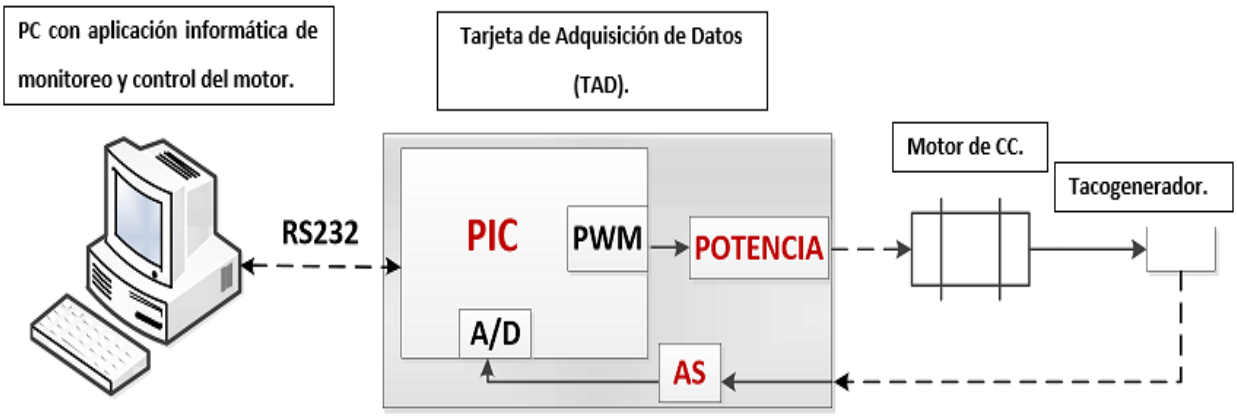


Figura 7. Esquema del sistema de control.

En el PIC se resalta un modulador de ancho de pulso (PWM) que controla la cantidad de energía que se envía al circuito de potencia para regular la velocidad del motor; y un conversor análogo digital (A/D) de 12 bits, que convierte la señal analógica proveniente del AS en digital. Este último básicamente es un divisor de tensión que acondiciona la señal del sensor Tacogenerador de 0 a 14.57 volts a un rango de -5 a 5 volts para el A/D (21).

2.4.1. Detalles técnicos del proceso.

El motor de CC tiene incorporado un tacogenerador o sensor de velocidad, además de un mecanismo de freno que permite fijar cargas a momento constante.

La Tabla 1 muestra los detalles técnicos del motor de corriente directa de imanes permanentes.

Tabla 1. Detalles técnicos del motor.

Modelo	PIVT 6-25/3A
Fabricante	DynamoSliven
País	Bulgaria

Tensión Nominal	30V
Corriente Nominal	3A
Velocidad Nominal	3000 rpm(314.16 rad/s)
Torque o Momento Nominal	0.1Nm
Potencia	31.4 W
Momento Máximo	0.225 Nm
Velocidad de Vacío	4858 rpm (508.75 rad/s)
Resistencia de Armadura	2.55 □
Inductancia de Armadura	4.8 mH
Corriente Máxima	11.76 A
Momento de Inercia Nominal	$0.0125 *10^{-3}$ Kg/m^2
Fricción Viscosa	$0.5*10^{-3}$ $Kg\ m^2/S$
Constante Electromecánica	0.0589
Constante de tiempo Mecánica	25 ms
Constante de tiempo Eléctrica	2 ms
Constante de la Fuerza Contraelectromotriz	7.2V/1000 rpm
Constante de Par	0.072 Nm

La siguiente figura representa de manera sencilla el mecanismo de freno.

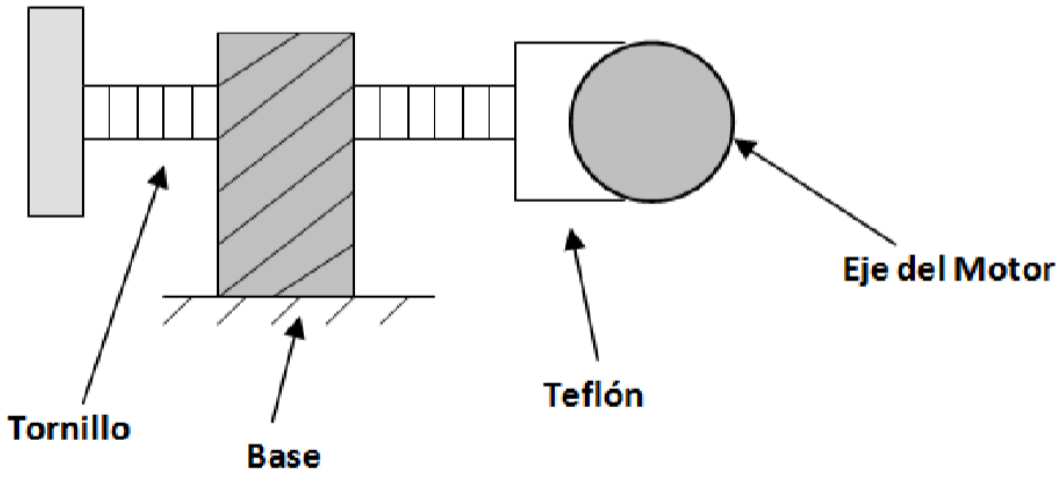


Figura 8. Mecanismo de freno.

El freno mecánico está compuesto por un tornillo terminado en una punta de teflón, la presión ejercida por la misma, sobre el eje del motor, aumenta a medida que se enrosque el tornillo sobre la pieza que lo sostiene.

La tabla que aparece a continuación refleja las características técnicas del sensor Tacogenerador.

Tabla 2. Características técnicas del sensor Tacogenerador.

Fabricante	DynamoSliven
Principio de funcionamiento	Electromagnético (Tacogenerador) (Ver principio de funcionamiento en (22).
Rango de medición	0-4858 rpm.
Alimentación	No necesita.
Ganancia	3V/1000 rpm
Rango de la salida	0-14.57 V
Precisión	5%.

2.5. Fase de exploración.

El ciclo de vida del proyecto se inicia con la fase de exploración, en ella el usuario asegura que exista suficiente material en las historias de usuario (HU) como para hacer una primera liberación del producto y los programadores se encargan de realizar las estimaciones correspondientes. El equipo de desarrollo se familiariza con cada una de las herramientas y tecnologías que se utilizará en el desarrollo del sistema y también explora diferentes posibilidades de conformar la arquitectura (23).

2.5.1. Listas de reservas del producto (LRP).

La lista de reserva del producto recoge todo el trabajo a realizar en el proyecto de manera priorizada. Esta lista se encuentra constantemente expuesta a cambios, puede crecer o modificarse según las nuevas necesidades del cliente y el conocimiento adquirido en el proceso de desarrollo, sin embargo solo se puede cambiar en cada iteración. La lista puede estar conformada por requerimientos técnicos y del negocio, funciones, errores a reparar, mejoras, defectos y actualizaciones tecnológicas.

Requisitos del sistema.

Los requisitos son la descripción de los servicios proporcionados por un sistema y sus restricciones operativas y reflejan las necesidades de los clientes que serán satisfechas con la implementación del sistema (24).

Requisitos funcionales.

Los requisitos funcionales describen lo que un sistema debe hacer. Dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque general tomado por la organización al redactar los requerimientos (24). A continuación se muestra el listado de requisitos:

RF 1: Modificar parámetros de configuración del puerto serie.

RF 2: Leer los valores de las variables del motor en tiempo real.

RF 3: Graficar los valores de las variables del motor en tiempo real.

RF 4: Modificar el valor de las pulsaciones del motor.

RF 5: Reiniciar sistema.

RF 6: Mostrar las muestras almacenadas.

RF 7: Guardar muestras en un fichero.

RF 8: Graficar muestras almacenadas.

RF 9: Cargar muestras desde fichero.

RF 10: Simular una muestra.

RF 11: Mostrar mensaje de alarma.

Requisitos no funcionales (RnF).

Los requisitos no funcionales son las cualidades o propiedades que debe poseer un sistema (24). A continuación se muestra el listado de requisitos no funcionales:

Tabla 3. Requisitos No Funcionales.

Apariencia o interfaz externa	**RNF #1:** La interfaz deberá ser sencilla de forma tal que el usuario no tenga que realizar muchas acciones para ejecutar una funcionalidad, dándole la opción de ejecutar dichas acciones en el menú o mediante un botón. **RNF #2:** El sistema propuesto contará con una barra de menú principal en la parte superior, donde el usuario puede realizar las diferentes opciones u operaciones que esta presenta. **RNF #3:** Todos los mensajes en pantalla aparecerán en idioma español.
Usabilidad	**RNF #4:** El usuario será informado a través de un mensaje en el momento en que ocurra un error en la entrada de los datos así como en la conexión al puerto serie. **RNF #5:** El sistema debe contar con menús que agrupen las funcionalidades que están relacionadas.
Software	**RNF #6:** Utilizar RTAI para la implementación en tiempo real. **RNF #7:** Utilizar *QtCreator* como entorno de desarrollo integrado. **RNF #8:** Utilizar *Visual* Paradigm para el modelado.

Hardware	**RNF #9:** Para alcanzar un buen desempeño con la aplicación la PC de trabajo debe tener como mínimo 1 GB de RAM y 40 GB de disco duro.
Portabilidad	**RNF #10:** El sistema operativo será GNU/Linux *Debian.*

2.5.2. Historias de usuario.

Una historia de usuario es una funcionalidad o parte de una funcionalidad en XP. Debe ser entendible por los clientes y el desarrollador, posible de probar, de valor para el cliente y lo suficientemente pequeña como para que el desarrollador pueda implementar media docena en una iteración (25).

Uno de los artefactos más importantes que genera la metodología XP son las Historias de Usuario. Estas tienen el mismo propósito que los casos de uso y son escritas por el desarrollador pero en constante diálogo y acuerdo con los clientes, tal y como ven ellos las necesidades del sistema, por tanto son descripciones cortas y escritas en el lenguaje del usuario sin terminología técnica. Conducen al proceso de creación de los test de aceptación, los cuales servirán para verificar que estas historias se han implementado correctamente. Otra de sus características es que solamente proporcionan los detalles sobre la estimación del riesgo y cuánto tiempo conllevará su implementación.

Las Historias de usuarios cuentan con la estructura que se explica a continuación:

Tabla 4. Muestra de una Historia de Usuario.

Historia de Usuario	
Número: (Número de la HU incremental en el tiempo)	**Nombre de Historia de Usuario:** (El nombre de la HU sería para identificarlas mejor entre los desarrolladores y el cliente)
Modificación de Historia de Usuario Número: (Si sufrió alguna modificación anterior)	
Usuario: (Se citan los desarrolladores responsables de la implementación de la HU)	**Iteración Asignada:** (Número de la iteración)
Prioridad en negocio: Grado de prioridad que le asigna el cliente a la HU en dependencia de sus necesidades. Los valores que puede tomar son	**Puntos estimados**: (El tiempo estimado en semanas que se demorará el desarrollo de la HU)

(Alta, Media o Baja).	
Riesgo en Desarrollo: Grado de complejidad que le asigna el equipo de desarrollo a la HU luego de analizarla. (Alto, Medio o Bajo).	**Puntos Reales:** (El tiempo que se demoró en realidad el desarrollo de la HU)
Descripción: (Breve descripción de la HU)	
Observaciones: (Señalamiento o advertencia del sistema)	
Prototipo de interfaz: (Prototipo de interfaz si aplica)	

A continuación se exponen algunas de las Historias de Usuario relacionadas con el sistema, el resto se pueden apreciar en el Anexo I:

Tabla 5. HU #1: Modificar parámetros de configuración del puerto serie.

Historia de Usuario	
Número: 1	**Nombre de Historia de Usuario:** Modificar parámetros de configuración del puerto serie.
Modificación de Historia de Usuario Número: -	
Usuario: Jimmy Bassa Martínez	**Iteración Asignada:** 1
Prioridad en negocio: Alto	**Puntos estimados**: 1
Riesgo en Desarrollo: Alta	**Puntos Reales:** 1
Descripción: Permite al usuario modificar los parámetros del puerto serie (puerto, velocidad, paridad, bits de datos, bits de parada, timeout y control de flujo) a través del cual se conecta la aplicación con el motor.	
Observaciones: En caso de que se ingresen valores no válidos o no se pueda conectar con el puerto serie seleccionado, el sistema muestra una advertencia al usuario.	
Prototipo de interfaz:	

Tabla 6. HU #4: Modificar el valor de las pulsaciones del motor.

Historia de Usuario	
Número: 4	**Nombre de Historia de Usuario:** Modificar el valor de las pulsaciones del motor.
Modificación de Historia de Usuario Número: -	
Usuario: Jimmy Bassa Martínez	**Iteración Asignada: 2**
Prioridad en negocio: Alto	**Puntos estimados**: 3
Riesgo en Desarrollo: Alto	**Puntos Reales:** 3
Descripción: Permite al usuario modificar el valor de la variable pulsación en el motor en tiempo real.	
Observaciones: En caso de no existir conexión serie con el motor el sistema muestra una advertencia.	
Prototipo de interfaz:	

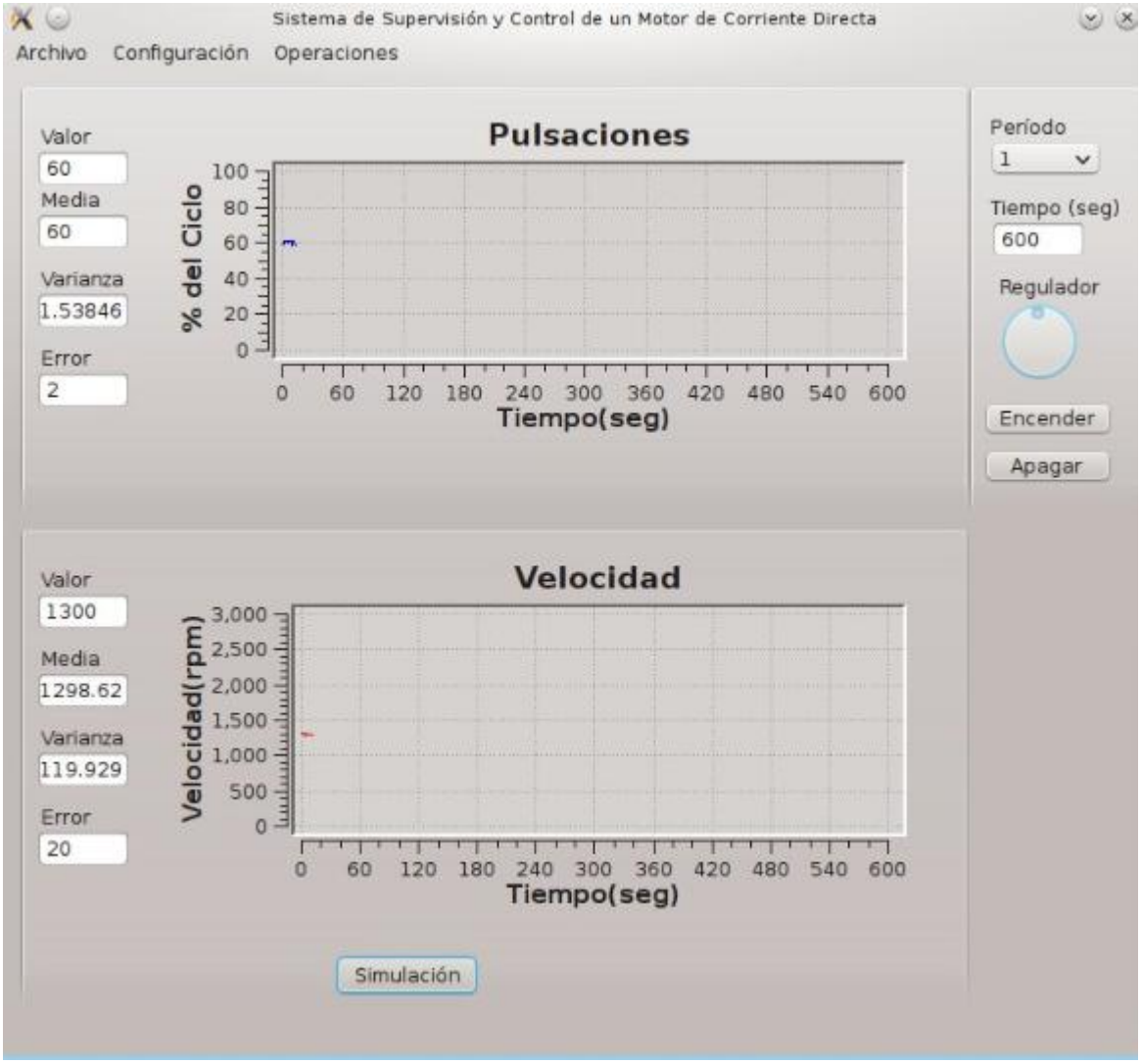
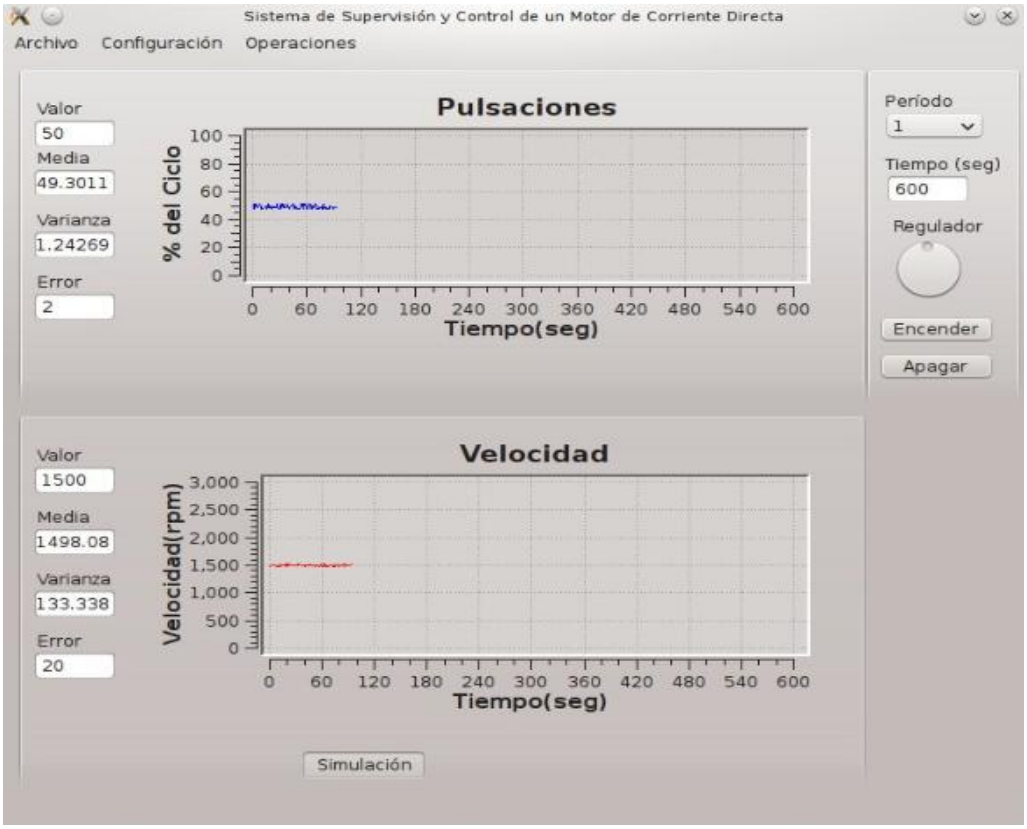


Tabla 7. HU #10: Simular una muestra.

Historia de Usuario	
Número: 10	**Nombre de Historia de Usuario:** Simular una muestra.
Modificación de Historia de Usuario Número: -	

Usuario: Jimmy Bassa Martínez	**Iteración Asignada:** 4
Prioridad en negocio: Bajo	**Puntos estimados**: 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1
Descripción: Permite al usuario simular una muestra a partir de valores entrados manualmente, sin realizar la lectura por el puerto serie.	
Observaciones:	
Prototipo de interfaz: 	

2.6. Fase de planificación.

El propósito de esta fase es llegar a un acuerdo entre clientes y desarrolladores sobre la posible fecha de entrega de cada grupo de funcionalidades. A partir de la prioridad dada por el usuario a cada historia, los programadores realizan una estimación del esfuerzo necesario para realizar cada una de ellas. Se toma como unidad de medida el punto. Un punto se considera como una semana (6 días) ideal de trabajo donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción. Generalmente, las HU necesitan de una a tres semanas de desarrollo (23).

2.6.1. Estimación de esfuerzo por historia de usuario.

Tabla 8. Estimación de esfuerzo por historias de usuario.

Historias de Usuario	Puntos estimados
Modificar parámetros de configuración del puerto serie.	1
Leer los valores de las variables del motor en tiempo real.	2
Graficar los valores de las variables del motor en tiempo real.	1
Modificar el valor de las pulsaciones del motor.	3
Reiniciar Sistema.	1
Mostrar las muestras almacenadas.	1
Guardar muestras en un fichero.	1
Graficar muestras almacenadas.	1
Cargar muestras desde fichero.	1
Simular una muestra.	1
Mostrar mensaje de alarma.	1

2.6.2. Plan de iteraciones.

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación.

De igual forma, para cada historia de usuario se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores. Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir (26).

Iteración 1: Se realizarán las historias de usuarios definidas dándole un enfoque al sistema en su etapa inicial. HU #1, HU #2.

Iteración 2: Se implementarán las historias de usuarios para el funcionamiento del sistema, dando una idea de cómo va avanzando la aplicación, la cual sirve como base para captar nuevos requerimientos y por último se realizarán las pruebas pertinentes a las funcionalidades implementadas. HU #3, HU #4.

Iteración 3: Se implementarán las historias de usuarios de mediana prioridad, se corregirán los errores o disconformidades encontradas en la iteración anterior. Se probarán cada una de las funcionalidades implementadas. HU #6, HU #7, HU #8, HU #9, HU #10.

Iteración 4: Se implementaran las historias de usuarios restantes. Al finalizar esta iteración se contará con una versión estable del producto con todos los requisitos implementados y probados. HU #5, HU #11.

Tabla 9. Plan de duración de las iteraciones.

Iteración	Orden de las Historias de Usuarios a implementar.	Duración (semanas)
1	Modificar parámetros de configuración del puerto serie. Leer los valores de las variables del motor en tiempo real.	3

2	Graficar los valores de las variables del motor en tiempo real. Modificar el valor de las pulsaciones del motor.	4
3	Mostrar las muestras almacenadas. Guardar muestras en un fichero. Graficar muestras almacenadas. Cargar muestras desde fichero. Simular una muestra.	5
4	Reiniciar Sistema. Mostrar mensaje de alarma.	2

2.6.3. Plan de entrega.

El cronograma de entregas no es más que el resultado de una reunión entre todos los actores del proyecto. Este cronograma se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores. Luego de algunas iteraciones es recomendable realizar nuevamente una reunión con los actores del proyecto, para evaluar nuevamente el plan de entregas y ajustarlo si es necesario.

Tabla 10. Plan de entrega.

Entregable	**Final 1ra Iteración 2da Semana de Marzo**	**Final 2da Iteración 2da Semana de Abril**	**Final 3ra Iteración 1ra Semana de Mayo**	**Final 4ta Iteración 4ta Semana de Mayo**
Aplicación	Versión 1.0 de la aplicación	Versión 2.0 de la aplicación	Versión 3.0 de la aplicación	Entrega final

2.7. Patrones de diseño.

"Un **patrón** es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos" (27).

"Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software." En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares (28).

2.7.1. Patrón de arquitectura Modelo-Vista-Controlador (MVC).

Es un patrón de arquitectura de software que separa la representación de la información a partir de la interacción del usuario con él. El modelo consta de datos de aplicaciones y reglas de negocio, y la entrada de controlador de media, convirtiéndolo a los comandos para el modelo o la vista. Una vista puede ser cualquier representación de salida de datos, como un gráfico o un diagrama. Múltiples vistas de los mismos datos son posibles, como un gráfico circular para la gestión y una vista tabular para los contadores. Las ideas centrales detrás de MVC son reutilización del código y la separación de preocupaciones (29).

Modelo: Esta es la representación específica de la información con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.

Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador: Este responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista.

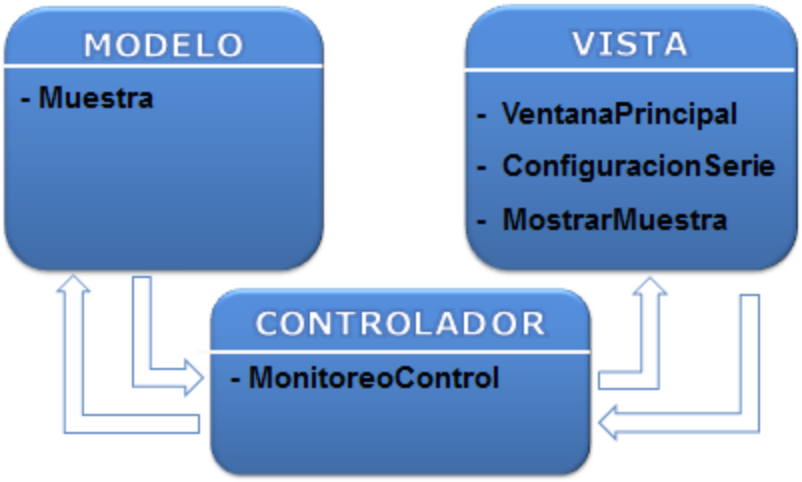


Figura 9. Representación del patrón MVC.

2.7.2. Patrones GRASP[4].

Los patrones **GRASP** describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos (30).

Los patrones GRASP se clasifican en cinco tipos, que serán descritos a continuación.

Experto: Es el encargado de establecer las responsabilidades a la clase que contiene la información necesaria para el cumplimiento de estas. Consiste en el principio fundamental de la asignación de responsabilidades en el diseño orientado a objetos. Ejemplo de utilización de este patrón lo podemos presenciar en la clase *ConexionSerie* la cual tiene la información y por tanto la responsabilidad de la conexión con el puerto serie, además del envió y recibo de una cadena mediante dicho puerto.

Creador: Es el encargado de asignar a una clase la responsabilidad de crear una instancia de otra clase, es decir la clase a la que se le asigna la responsabilidad es el creador de los objetos de la clase instanciada. Su principio fundamental es guiar la asignación de responsabilidades relacionadas con la creación de objetos. Se pueden evidenciar en la *clase MonitoreoControl* que es la responsable de crear una instancia de la *clase ConexionSerie*.

[4] **GRASP:** es un acrónimo que significa General Responsibility Assignment Software Patterns (en español, Patrones Generales de Software para Asignar Responsabilidades).

Controlador: Funciona como intermediario entre una interfaz y el algoritmo que la implementa, de forma tal que es quien recibe los datos del usuario y quien los envía a las distintas clases según el método que se llame. Un controlador es un objeto de interfaz no destinada al usuario encargado de manejar un determinado evento del sistema. En nuestra solución un ejemplo de la utilización de este patrón es la *clase MonitoreoControl* la cual se encarga de delegar a los objetes existentes en ella de las Clase ConexiónSerie y Muestras el trabajo que ha de realizarse.

Alta cohesión: Una clase almacena información coherente que esté en la mayor medida posible relacionada con otra clase. Clases con alta cohesión comparten las responsabilidades de las operaciones entre ellas. Aunque este patrón se ve reflejado en todas las clases, un ejemplo de esto es la clase ConexionSerie que es la única clase dentro de la solución que se encarga del envió y la recepción de datos por el puerto serie. Aunque podemos decir que cada una de las clases en el sistema desempeñan una tarea específica que no realizan las demás.

Bajo acoplamiento: Este patrón se encarga de tener las clases lo menos ligadas entre sí, de manera tal que si ocurren modificaciones en alguna, repercuta lo menos posible en las demás favoreciendo la reutilización y reduciendo la dependencia entre clases. Este patrón se ve reflejado en todas las clases de la aplicación tratando de que exista la menor dependencia posible entre las clases existentes.

2.7.3. Patrones GOF.

Los Patrones GOF por sus siglas en inglés (Gang of Four, Banda de Cuatro) proporcionan a los programadores una estructura de código común a todos los proyectos que implemente una funcionalidad genérica. La utilización de estos patrones de diseño, permite ahorrar tiempo en la construcción del software, así como hacerlo más fácil de comprender, mantener y extender (29).

Durante el diseño del sistema se emplearon patrones GOF como:

Singleton (instancia única): Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella (31).

El patrón Singleton se pone de manifiesto en la solución en la clase ConexionSerie debido a que es conveniente asegurar la existencia de una única instancia de dicha clase.

2.8. Tarjetas Clase – Responsabilidad – Colaborador (CRC).

Las tarjetas CRC, propias de la metodología XP, son una técnica simple e informal pero efectiva que ha sido propuesta para el diseño detallado de sistemas Orientado a Objetos. Se analizan basándose en sus responsabilidades con respecto al sistema y permiten que el equipo completo contribuya en la tarea del diseño. Una tarjeta CRC representa un objeto, el nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda, y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente.

Las responsabilidades son los atributos y las operaciones relevantes para la clase. Dicho de una manera más simple una responsabilidad es “cualquier cosa que la clase sabe o hace”. Los colaboradores son aquellas clases que se requieren para que una clase reciba la información necesaria para completar una responsabilidad (32).

A continuación se muestra una de las tarjetas del sistema a desarrollar, el resto se encuentra en el Anexo II:

Tabla 11. Clase MonitoreoControl.

Tarjeta CRC	
Datos de la clase	
Nombre de la clase: MonitoreoControl	
Descripción: Es la encargada de controlar el proceso de conexión al puerto serie, la lectura y escritura de parámetros.	
Responsabilidades	**Colaboradores**
conectarse leerDatos	ConexionSerie

adicionarMuestra salvarMuestra cargarMuestra	Muestra

2.9. Consideraciones finales.

En este capítulo se definieron las características y funcionalidades del sistema, se reflejaron los requisitos funcionales en forma de Historias de Usuario y no funcionales que darán una eficiente implementación y diseño a la aplicación. Se describieron las historias de usuarios asociadas a sus respectivas tareas de ingeniería que se deben desarrollar con el objetivo de complementar cada iteración del ciclo del proyecto. Además se definió el plan de entrega el cuál será guía para la creación del cronograma de trabajo, y se definieron las tarjetas CRC con el objetivo de lograr una mejor compresión de la solución propuesta.

Se concluye, además, que todos los artefactos generados por la metodología en esta etapa guiarán de manera efectiva el desarrollo de la herramienta.

Capítulo 3: Implementación y prueba.

3.1. Introducción.

En el presente capítulo se abordan los temas relacionados con la implementación y pruebas realizadas al sistema. Además se organizan las clases fundamentales del sistema. Se procede a desarrollar las tareas de la ingeniería que corresponden a las HU desarrolladas anteriormente, luego mediante las pruebas se verifica que el producto resultante cumpla con los requisitos definidos. En esta etapa se detectan y corrigen errores para la posterior aceptación del producto.

3.2. Implementación de la aplicación.

Para la implementación de la solución se tiene como meta, alcanzar el desarrollo total de la misma, así como para su mejor entendimiento una correcta organización del código en aras de funcionar como un todo. Para estas se desglosan las HU en tareas de ingeniería, las cuales guían la implementación, por lo que es más factible el desarrollo del sistema logrando una programación eficiente.

3.2.1. Tareas de ingeniería.

Las Tareas de Ingeniería (TI) se usan para describir las tareas que se realizan sobre el proyecto, estas pueden ser de desarrollo, corrección o mejora. Las TI tienen relación con una historia de usuario. Estas están compuestas por la fecha de inicio y fin, el programador responsable de cumplirla estima el tiempo que tardará en realizar cada una de las tareas y se realiza una pequeña descripción de lo que se tratará de hacer en la tarea.

A continuación se exponen las primeras cinco tareas de ingeniería correspondientes a las HU representadas anteriormente, las restantes aparecen en el Anexo II:

Tabla 12 - Tarea # 1. Modificar parámetros de configuración del puerto.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1

Nombre Tarea: Modificar parámetros de configuración del puerto	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3 días
Fecha Inicio: 5/2/2014	**Fecha Fin:** 8/2/2014
Programador Responsable: Jimmy Bassa Martínez	
Descripción: Para modificar los parámetros de configuración del puerto serie el usuario deberá seleccionar en la barra de menú la operación configurar parámetros del puerto serie, y esta visualizará los respectivos parámetros a configurar para la conexión serie.	

Tabla 13. Tarea # 2. Encender motor de corriente.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 2
Nombre Tarea: Encender el motor de corriente.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3 días
Fecha Inicio: 18/2/2014	**Fecha Fin:** 21/2/2014
Programador Responsable: Jimmy Bassa Martínez	
Descripción: Para realizar esta tarea se establece la conexión con el puerto serie y se le envía un valor de pulsaciones mayor que cero para el encendido del motor.	

Tabla 14. Tarea #3. Leer variables del motor

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 2
Nombre Tarea: Leer variables del motor.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3 días

Fecha Inicio: 1/3/2014	**Fecha Fin:** 4/3/2014
Programador Responsable: Jimmy Bassa Martínez	
Descripción: Para la realización de esta tarea el usuario deberá acceder a la barra de menú y ejecutar la operación leer valores de las variables del motor, y se verán dichos valores en el periodo e intervalo de tiempo establecido.	

Tabla 15. Tarea # 4. Graficar el valor de la variable pulsaciones del motor.

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 3
Nombre Tarea: Graficar el valor de la variable pulsaciones del motor.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3 días
Fecha Inicio: 7/3/2014	**Fecha Fin:** 10/3/2014
Programador Responsable: Jimmy Bassa Martínez	
Descripción: Se grafican los valores leídos en función del tiempo.	

Tabla 16. Tarea # 5. Graficar el valor de la variable velocidad del motor.

Tarea de Ingeniería	
Número Tarea: 5	**Número Historia de Usuario:** 23
Nombre Tarea: Graficar el valor de la variable velocidad del motor.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3 días
Fecha Inicio: 12/3/2014	**Fecha Fin:** 15/3/2014
Programador Responsable: Jimmy Bassa Martínez	

Descripción: Se grafican los valores leídos en función del tiempo..

3.2.2. Estándares de codificación.

Se le llama estándares de codificación a los convenios que existen para escribir el código fuente con un formato estándar para todos los programadores. Aseguran la legibilidad del código y facilitan el traceo del mismo. Los estándares de codificación permiten entender, de manera rápida el código empleado en el desarrollo de una aplicación. El uso de las técnicas de codificación tiene una gran importancia para realizar buenas prácticas de programación y escribir un código de alta calidad. Es por ello que para la implementación del sistema se definen los requisitos siguientes:

Indentación:

La unidad de indentación de bloques de sentencias son 4 espacios.

```
32  double MonitoreoControl::media(){
33      double suma = 0;
34      for(int i= 0; i < pos; i++ ){
35          suma += y[i];
36      }
37      double prom = (double)(suma/(pos));
38      return prom;
39  }
```

Se muestra un fragmento de código: Ejemplo de indentación.

Declaración de variables:

Se debe declarar cada variable en una línea distinta, de esta forma cada variable se puede comentar por separado.

```
18      QString  fecha; //fecha en que se realizala muestra.
19      QString hora;   //hora en que  se realiza la muestra.
20      int cantReal;   //cantidad de valores de la muestra.
```

Se muestra un fragmento de código: Ejemplo de declaración de variables.

Declaración de métodos:

No hay espacio entre el nombre del método, el paréntesis y la lista de parámetros. Se abre la llave "{" al final de la misma línea que la declaración.

La llave de"}" debe aparecer en una línea aparte con la misma indentación que el método o clase que cierra.

```
4  void ConexionSerie::enviarMensaje(QString mensaje){
5      serialPort->write(mensaje.toAscii(),mensaje.length());
6  }
```

Se muestra un fragmento de código: Ejemplo de declaración de métodos.

Nombre de las clases:

Las clases comenzarán con mayúscula, si la clase es compuesta empezarán con mayúsculas las dos letras iniciales de cada palabra.

Sentencias simples:

Cada línea debe contener una sola sentencia.

```
51 double MonitoreoControl::varianza(){
52     double suma = 0;
53     double prom = media();
54     for(int i= 0; i < pos; i++ ){
55         suma+= ((y[i]-prom)*(y[i]-prom));
56     }
57     double var = (double)(suma/(pos));
58     return var;
59 }
```

Se muestra un fragmento de código: Ejemplo de sentencias simples.

Sentencia de retorno:

No se pondrá la expresión de retorno entre paréntesis a menos que con ello se gane en claridad.

Sentencia if, for, while:

Deben estar indentadas a un nivel superior que el precedente. Todas las sentencias del tipo if o for, while, deberán tener llaves aunque sólo contengan una sentencia, de esta forma se evita la introducción accidental de errores si se añaden posteriormente sentencias.

3.3. Pruebas de software.

La metodología XP propone la constante realización de pruebas, los desarrolladores dirigen el trabajo a la búsqueda de errores, someten sistemáticamente a pruebas las funcionalidades del sistema lo que permite aumentar la calidad de estos reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección.

XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñada por el cliente (33). Al sistema se le realizaron las pruebas unitarias y las pruebas de aceptación verificando que cumpla con los requisitos definidos por el cliente.

3.3.1. Pruebas unitarias.

Las pruebas unitarias son una de las piedras angulares de XP. Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados.

En el transcurso de la implementación del sistema cada uno de los desarrolladores prueba constantemente lo que va obteniendo para garantizar que las funcionalidades exigidas por el cliente estén siendo implementadas correctamente. Cuando se encuentra un error, éste debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir.

Para desarrollar dicha tarea se comprueban los caminos lógicos del sistema mediante casos de prueba, que comprueben los algoritmos implementados, por lo que se utilizó la técnica Camino básico junto con la métrica Complejidad ciclomática.

Se seleccionaron dos métodos para aplicarle la técnica antes mencionada. El primer método seleccionado lleva como nombre *auxFlowControl* que dado un valor entero entre cero y dos retorna el valor del enumerativo definido para el control de flujo.

```
80  FlowType  ConfiguracionSerie::auxFlowControl(int indice)
81  {
82       if (indice == 0)
83            return FLOW_OFF;
84       else
85          if (indice == 1)
86             return FLOW_HARDWARE;
87          else
88               return FLOW_XONXOFF;
89
90  }
```

Figura 10. Prueba unitaria al método *auxFlowControl*.

Para obtener los casos de prueba a partir de este algoritmo se debe construir inicialmente el grafo de flujo correspondiente al código.

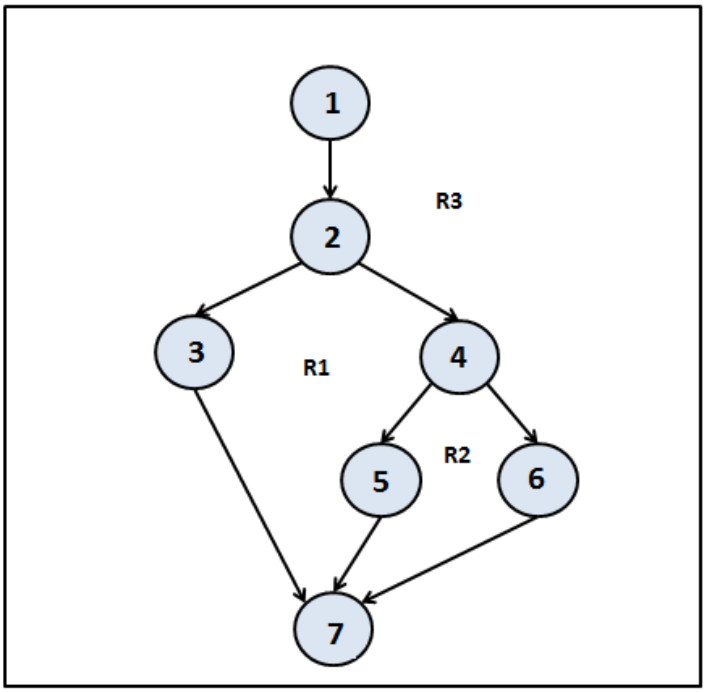


Figura 11. Prueba unitaria Grafo de flujo.

Luego se determina la complejidad ciclomática V (G) del grafo resultante G, para esto se utiliza la siguiente fórmula:

$$\boldsymbol{V(G) = A - N + 2}$$

Donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

$$\boldsymbol{V(G) = A - N + 2}$$

$$\boldsymbol{V(G) = 8 - 7 + 2}$$

$$\boldsymbol{V(G) = 3}$$

El número de caminos independientes de la estructura del programa es igual a 3, y los caminos independientes son:

Camino 1: 1, 2, 3, 7.

Camino 2: 1, 2, 4, 5, 7.

Camino 3: 1, 2, 4, 6, 7.

Además el número de regiones del grafo de flujo coincide con la complejidad ciclomática.

El segundo método es *clicDerecho* que realiza la graficación de una muestra seleccionada por el usuario en la tabla.

```
34  void MostrarMuestras::clicDerecho(const QPoint &pos)
35  {
36      QMenu* menu = new QMenu(ui->tableWidget);
37      QTableWidgetItem* item = ui->tableWidget->itemAt(pos.x(), pos.y());
38      if (item)
39      {
40          switch (item->type())
41          {
42          case 1:
43              menu->addAction("Graficar");
44              connect(menu->actions().at(0),SIGNAL(triggered()),this,SLOT(Graficar()));
45              menu->exec(ui->tableWidget->mapToGlobal(pos));
46              break;
47          }
48      }
49  }
```

Figura 12. Prueba unitaria al método *clicDerecho*.

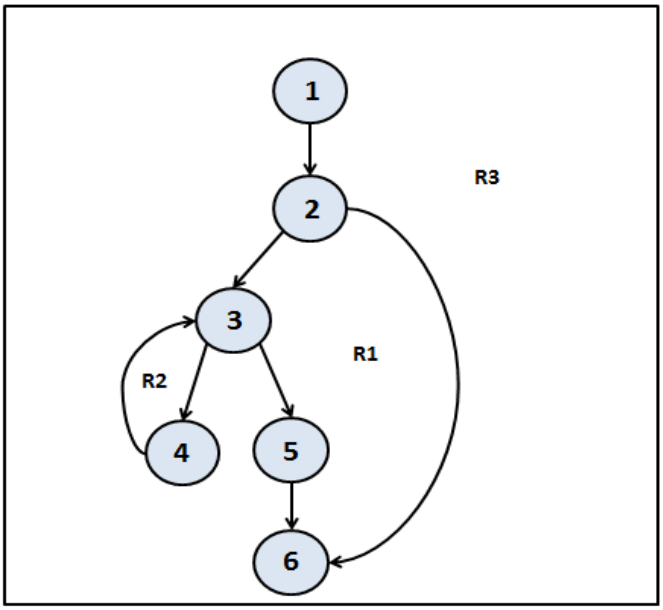


Figura 13. Prueba unitaria Grafo de flujo.

$$\boldsymbol{V(G) = A - N + 2}$$

$$\boldsymbol{V(G) = 7 - 6 + 2}$$

$$\boldsymbol{V(G) = 3}$$

Camino 1: 1, 2, 3, 4, 3, 5, 6.

Camino 2: 1, 2, 3, 5, 6.

Camino 3: 1, 2, 6.

Análisis de los resultados

Para la validación del código en el desarrollo del sistema se seleccionaron los métodos que se consideraron más importantes, realizándole las pruebas en función de evaluar si el funcionamiento de cada uno de los métodos desarrollados se comportó de manera correcta. Se ejecutaron un total de 18 pruebas a los 10 métodos seleccionados, de las cuales 14 resultaron satisfactorias en una primera iteración de pruebas y 4 no satisfactorias, solucionándose luego en una segunda iteración, con la realización de estas pruebas se logró comprobar la estabilidad de la lógica aplicada en el código generado en el desarrollo del sistema.

3.3.2. Pruebas de aceptación.

Las pruebas de aceptación son creadas en base a las historias de usuarios. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. Estas pruebas se enfocan en las características y las funcionalidades del sistema, son las encargadas de comprobar que las funcionalidades desarrolladas sean las esperadas por el cliente. Luego de haber superado las pruebas de aceptación podrá considerarse que la aplicación es apta para el uso y despliegue dentro del proyecto.

A continuación se muestran algunos casos de pruebas de aceptación correspondientes a las HU, el resto se puede apreciar en el Anexo IV:

Tabla 17. Caso de prueba de aceptación: Configuración del puerto serie.

Caso de Prueba de Aceptación	
Código: HU1p1	HU1: Modificar parámetros de configuración del puerto serie.
Nombre: Configuración del puerto serie.	
Descripción: Se desea probar que el sistema permita modificar los parámetros de configuración del puerto serie para poder conectarse al motor de corriente, en caso de no poder conectarse se mostrará un mensaje de advertencia indicando que no se pudo conectar al puerto serie.	
Condiciones de ejecución: Juego de datos correctos.	
Entrada/ Pasos de ejecución: Acceder en el menú operaciones a la opción Modificar parámetros de configuración del puerto serie. Seleccionar las diferentes opciones de configuración para conectarse al puerto serie y dar click en el botón aceptar.	
Resultado esperado: Conexión con el puerto serie.	
Evaluación de la prueba: Aceptada.	

Tabla 18. Caso de prueba de aceptación: Leer los valores en tiempo real.

Caso de Prueba de Aceptación	
Código: HU2p1	HU2: Leer los valores de las variables del motor en tiempo real.
Nombre: Leer los valores en tiempo real.	
Descripción: Se desea probar que el sistema permite leer los valores de las variables del motor de corriente (pulsaciones y velocidad) en tiempo real, para graficar el comportamiento de las mismas en la aplicación. Para realizar esta operación es necesario estar conectado al puerto serie de la TAD. En caso de no existir conexión se debe mostrar un mensaje diciendo que no se puede realizar la lectura, y se le pregunta al usuario si desea realizar una simulación.	
Condiciones de ejecución: Existir conexión con el motor.	
Entrada/ Pasos de ejecución: Acceder a la opción Leer los valores de las variables del motor en tiempo real que se encuentra en la barra de menú.	
Resultado esperado: Los valores de las variables de las pulsaciones y velocidad del motor de corriente directa se leen y almacenan en tiempo de ejecución en el sistema en tiempo real.	
Evaluación de la prueba: Aceptada.	

Resultados esperados.

El siguiente gráfico muestra el resumen del resultado de las pruebas realizadas.

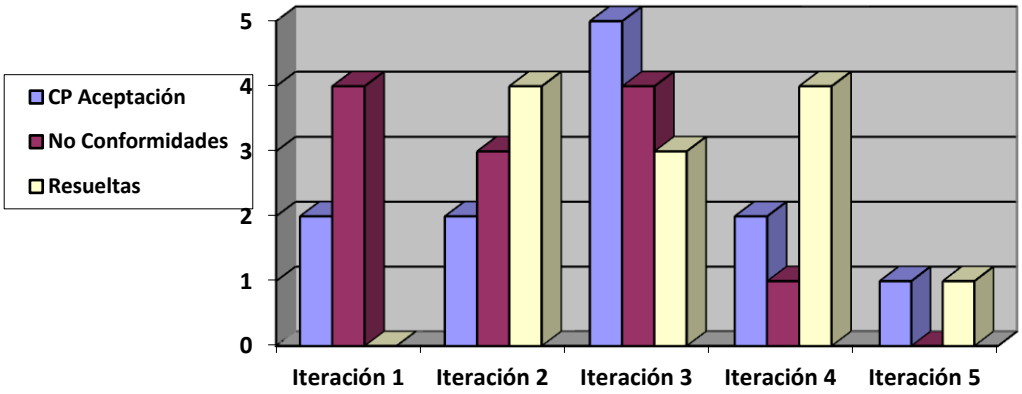


Figura 14. Resultado de las pruebas realizadas.

En la gráfica se pueden evidenciar los resultados de las pruebas de aceptación, donde para una primera iteración se realizaron dos casos de pruebas de aceptación correspondientes a las HU#1 y HU#2, encontrándose cuatro no conformidades, siendo estas una de error ortográfico, dos de validación y una de interfaz. Para una segunda iteración a dos casos de pruebas correspondientes a las HU#3 y HU#4 se encontraron tres no conformidades, una de validación y dos de interfaz, quedando resueltos los errores que se detectaron al inicio. En la tercera iteración se realizaron cinco casos de pruebas correspondientes a las HU#6, HU#7, HU#8, HU#9 y HU#10, produciéndose cuatro no conformidades, dos de validación y dos de ortografía, quedando resueltas las tres no conformidades de la iteración anterior. En la cuarta iteración se realizaron dos casos de pruebas correspondientes a las HU#5 y HU#11 donde se encontró una no conformidad de interfaz. Finalmente se realizó una última iteración para un caso de prueba de aceptación y se concluyó que todos los requisitos del sistema funcionan correctamente y que el software no presenta errores.

3.4. Consideraciones finales.

En el siguiente capítulo quedo reflejado el uso de los estándares de codificación, evidenciando el uso de las buenas prácticas de programación. Se desarrolló además, un proceso para la validación de la solución del software, el cual consistió en la realización de pruebas unitarias y pruebas de aceptación. De esta manera el resultado de la validación demostró que las funcionalidades de la aplicación son completamente operativas y producen el resultado esperado, por lo que se cumple con los objetivos que se perseguían con su desarrollo. Se demuestra con lo planteado, que el sistema queda listo para

realizar correctamente el control y monitoreo del motor de corriente directa en tiempo real, mediante una aplicación de escritorio.

Conclusiones generales

El desarrollo del presente trabajo estuvo basado en un conjunto de etapas que fueron ejecutadas para dar cumplimiento al objetivo general del mismo en pos de alcanzar las tareas propuestas para su realización. Además permitió arribar a las siguientes conclusiones:

- ✓ Se completó el estudio del arte referente a los sistemas de monitoreo y control, conceptualizándose las principales características de la aplicación a desarrollar; siendo una de las más importante la posibilidad de que los usuarios visualicen en tiempo real las mediciones de las variables (pulsaciones y velocidad) del motor de corriente directa.

- ✓ Se implementó una aplicación de escritorio que permite el control y la supervisión de dispositivos en tiempo real, utilizando para su realización el sistema operativo Debian y RTAI para tener control del tiempo en que se ejecutan las tareas.

- ✓ Se validó funcionalmente la aplicación desarrollada mediante una estrategia de pruebas unitarias y pruebas de aceptación, con lo cual se comprobó que el sistema no contiene errores de implementación y cumple con los requisitos funcionales planteados por el cliente.

- ✓ Se validó cualitativamente el sistema por parte del cliente, quien emitió como constancia de satisfacción un aval de cumplimiento de los objetivos trazados.

Recomendaciones

El objetivo general del presente trabajo de diploma fue cumplido satisfactoriamente, pero a fin de enriquecer el trabajo realizado han surgido algunas ideas recomendables para su futuro perfeccionamiento:

- ✓ Realizar un análisis estadístico profundo de las muestras almacenadas.
- ✓ Agregar nuevas funcionalidades que permitan la lectura y el control de las restantes variables del motor.

Referencias bibliográficas

1. *Sistema de monitoreo y control en tiempo real para un horno.* **Cruz, Frank Pacareo de la and Fernández Pedroso, Elier.** La Habana : s.n., 2012.

2. **Instituto de Cibernética, Matemática y Física.** ICIMAF. [En línea] 2010. [Citado el: 15 de Noviembre de 2013.]

3. **Wordpress.** Definición de. [En línea] 2008. [Citado el: 25 de Noviembre de 2013.] definicionde.de.

4. **Cabrera, Elibeth.** Monografías. [En línea] 2008. [Citado el: 25 de Noviembre de 2013.] http://www.monografias.com/trabajos14/control/control.shtml#co.

5. *Sistemas de tiempo real.* **Ortiz, Dilcia Orebelina Elvir.** México : s.n., 27 de Septiembre de 2010.

6. *Sistemas Operativos de Tiempo Real.* **Vignoni, Ing. José Roberto.** Argentina : s.n., 2004.

7. *Una plataforma tiempo real para ejecutar algoritmos de control.* **Mejías, Carlos, León, Leandro y Ríos, Addison.** 2, Venezuela : s.n., 2010, Vol. 31. ISSN 1316-7081.

8. *Control de una Plataforma Giroestabilizada en Tiempo Real.* **Aciti, Claudio y Acosta, Nelson.** [ed.] Fac. de Ciencias Exactas- Universidad Nacional del Centro de la Prov. de Bs. As. Argentina : s.n.

9. *RTAI – Real Time Application Interface.* **Acosta., Francisco Resquín.** Paraguay : s.n., 7 de Septiembre de 2009.

10. **Mejías, Juan Camilo Restrepo.** ISSUU. [En línea] 2012. [Citado el: 5 de Diciembre de 2013.] http://issuu.com/camilorestrepomejia/docs/sistemas_empotrados.

11. **Henriquez, Carlos.** Sistemas embebidos. [En línea] 16 de Febrero de 2008. [Citado el: 12 de Diciembre de 2013.] http://sistemasembebidosarquitectura.blogspot.com/200..

12. *Máquinas eléctricas.* **Feito, Javier Sanz.** Madrid : s.n., 2010.

13. **Fritz, Caroline.** EHow en español. [En línea] [Citado el: 12 de Diciembre de 2013.] http://www.ehowenespanol.com/definicion-motor-corriente-directa-sobre_55810/.

14. **Instituto Profesional Ingeniería en Automatización y Control Industrial.** Santo Tomás. [En línea] 2010. [Citado el: 16 de Diciembre de 2013.] https://www.santotomas.cl/areas/areas/detalleCarrera/ip/ingenieria/176/ip-carrera-ingenieria-en-automatizacion-y-control-industrial.

15. Debian. [En línea] 2013. [Citado el: 16 de Diciembre de 2013.] http://www.debian.org/intro/about.es.html..

16. *Metodologías ágiles en el desarrollo de Software.* **José H Canó, Patricio Letelier, Mª Carmen Penadés.** España : s.n., 2003.

17. Buenas tareas. [En línea] [Citado el: 20 de Enero de 2014.] http://www.buenastareas.com/ensayos/Metodologias-Agiles-Rup/1223371.html..

18. [En línea] 2012. [Citado el: 21 de Enero de 2014.] http://pnfiingenieriadesoftwaregrupocuatro.blogspot.com..

19. **Qt Project Hosting.** [En línea] [Citado el: 20 de enero de 2014.] https://qt-project.org/..

20. [En línea] [Citado el: 20 de Enero de 2014.] http://www.visual-paradigm.com..

21. **Panadés, M. L.** Sistema para el control de la velocidad de un motor de corriente continua empleando técnicas estadísticas y de simulación. *Trabajo de Diploma.* s.l. : Instituto Superior Politécnico José Antonio Echeverría, 1999.

22. **Marin, L.** [En línea] 4 de Diciembre de 2011. [Citado el: 30 de Enero de 2014.] http://miluisyedgar.blogspot.com/2007/05/sensortacogenerador.html.

23. **Beck, Kent.** *Extreme Programming Explained: Embrace Change.* Estados Unidos de América. : s.n., 200. ISNB 201616416.

24. **Sommerville, I.** *Ingeniería del Software.* Madrir : s.n., 2005. págs. 108-121.

25. **Beck, Kent.** *Planning Extreme Programming.* Estados Unidos de América : s.n., 2001. ISBN 201710919.

26. *Reglas y prácticas en Xtreme Programming.* **Joskowicz, Ing. José.** España : s.n., 2008.

27. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificadoc.* **Larman, C.** s.l. : Prentice Hall, 2006, págs. 162-185.

28. **Tedeschi, Nicolás.** Microsoft developers Netrwork. [En línea] [Citado el: 29 de Marzo de 2014.] http://msdn.microsoft.com/es-es/library/bb972240.aspx..

29. *CONSECUENCIAS DE LA PRESTACIÓN DE SERVICIOS ADICIONALES:* **Rosario López Gavira, José Angel Pérez López.** España : s.n.

30. **Grosso, A.** Prácticas de software. [En línea] 21 de Marzo de 2011. [Citado el: 29 de Marzo de 2014.] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

31. **Prieto, Félix.** [En línea] 2009. [Citado el: 29 de Marzo de 2014.] http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf..

32. **Roger S Pressman.** Ingeniería del software. Un enfoque práctico. 6ta. 2006, 8 Modelado de análisis.

33. *PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA.* **http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf.** España : s.n.

Bibliografía consultada

1. **Navarro, Giovanni, Chacón, Diana y Jeréz, Camilo.** [En línea] 15 de Marzo de 2013. [Citado el: 26 de Noviembre de 2013.] http://prezi.com/p_mqff05fnct/definicion-de-los-sistemas-de-monitoreo-en-sistemas-de-gestion-integral-de-energia/.

2. **Salazar, Fabritsio.** ElectroIndustria. [En línea] Octubre de 2008. [Citado el: 30 de Noviembre de 2013.] http://www.emb.cl/electroindustria/articulo.mvc?xid=1063&edi=56.

3. **Milburn, Jaqueline Berumen.** *Monitoreo y control de proyectos.* Colombia : Pregón Ltda, 2010. ISBN:978-958-8590-10-3.

4. **Comité de Normalización de Petróleos Mexicanos y Organismos Subsidiarios.** *SISTEMAS DIGITALES DE MONITOREO Y CONTROL.* México : s.n., 2012.

5. **Báez, Ing Jesús.** SISTEMA CENTRAL DE MONITOREO Y CONTROL REMOTO PARA EQUIPOS DE AIRE ACONDICIONADO. [En línea] [Citado el: 13 de Diciembre de 2013.] http://www.acee.com.mx/pdf/monitoreo%20de%20aires%20acondicionados.pdf.

6. *Sistema de riego automatizado en tiempo real con balance hídrico, medición de humedad del suelo y lisímetro*.* **Martiniano Castro Popoca, Francisco Miguel Águila Marín.** 2, México : s.n., 2008, Vol. 34. ISSN 0568-2517.

7. **COMBIOMED Equipos Médicos.** COMBIOMED. [En línea] 2010. http://www.combiomed.sld.cu/descargas/sueltos/es/MOVICORDE%20Catalogo%20ESPANOL.pdf.

8. *Sistema de Control y Monitoreo Automatizado.* **Ing. Carlos Davison Aguiar, Ing. Anabel Zuaznabar Horta.** La Habana : s.n., 2012.

9. La revista informática. com. [En línea] 2006. [Citado el: 16 de Diciembre de 2013.] http://www.larevistainformatica.com/C1.htm.

Glosario de términos

- ✓ **API:** Interfaz de Programación de Aplicaciones.
- ✓ **AS:** Acondicionador de Señal.
- ✓ **CITMA:** Ministerio de Ciencia, Tecnología y Medio Ambiente.
- ✓ **CRC:** Tarjetas Clase – Responsabilidad – Colaborador.
- ✓ **DCS:** Sistemas de Control Distribuido.
- ✓ **GPS:** Sistema de Posicionamiento Global.
- ✓ **GRASP:** Patrón de Asignación de Responsabilidades.
- ✓ **HU:** Historias de Usuarios.
- ✓ **ICIMAF:** Instituto de Cibernética, Matemática y Física.
- ✓ **IPC:** Comunicación entre Procesos.
- ✓ **LRP:** Lista de Reserva del Producto.
- ✓ **MVC:** Patrón arquitectónico Modelo-Vista-Controlador.
- ✓ **OOP:** Programación Orientada a Objetos.
- ✓ **PIC:** Controlador de Interfaz Periférico.
- ✓ **PWM:** Modulador por Ancho de Pulsos.
- ✓ **RPC:** Llamadas de Procedimientos Remotos.
- ✓ **Rpm:** Revoluciones por minutos.
- ✓ **RUP:** Proceso Unificado de Desarrollo.

- ✓ **SCADA:** Sistemas de Adquisición de Datos y Control Supervisorio.
- ✓ **STR:** Sistemas en Tiempo Real.
- ✓ **SO:** Sistema Operativo.
- ✓ **SOTR:** Sistemas Operativos en Tiempo Real.
- ✓ **TAD:** Tarjeta de Adquisición de Datos.
- ✓ **TI:** Tareas de Ingeniería.
- ✓ **TIC:** Tecnologías de la Información y las Comunicaciones.
- ✓ **XP:** Programación Extrema.

Anexos

Anexo I – Historias de Usuario.

HU #2: Leer los valores de las variables del motor en tiempo real.

Historia de Usuario	
Número: 2	**Nombre de Historia de Usuario:** Leer los valores de las variables del motor en tiempo real.
Modificación de Historia de Usuario Número: -	
Usuario: Jimmy Bassa Martínez	**Iteración Asignada:** 1
Prioridad en negocio: Alto	**Puntos estimados**: 2
Riesgo en Desarrollo: Alta	**Puntos Reales:** 2
Descripción: Permite al usuario realizar una acción de lectura en tiempo real de los valores de las variables del motor.	
Observaciones: En caso de no existir conexión serie con el motor el sistema muestra una advertencia.	
Prototipo de interfaz:	

HU #3: Graficar los valores de las variables del motor en tiempo real.

Historia de Usuario	
Número: 3	**Nombre de Historia de Usuario:** Graficar los valores de las variables del motor en tiempo real.
Modificación de Historia de Usuario Número: -	

Usuario: Jimmy Bassa Martínez	**Iteración Asignada:** 2
Prioridad en negocio: Alto	**Puntos estimados**: 1
Riesgo en Desarrollo: Alta	**Puntos Reales:** 1
Descripción: Muestra al usuario una gráfica con la variación de los valores de las variables en el momento de lectura con respecto al tiempo.	
Observaciones: Esta operación ocurre al mismo tiempo en que se van leyendo los valores del motor.	
Prototipo de interfaz:	

HU #5: Reiniciar sistema.

Historia de Usuario	
Número: 5	**Nombre de Historia de Usuario:** Reiniciar sistema.
Modificación de Historia de Usuario Número: -	
Usuario: Jimmy Bassa Martínez	**Iteración Asignada:** 4
Prioridad en negocio: Bajo	**Puntos estimados**: 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1
Descripción: Permite establecer todas las configuraciones cambiadas a sus respectivos valores por defecto mediante el reseteado del sistema.	
Observaciones:	
Prototipo de interfaz:	

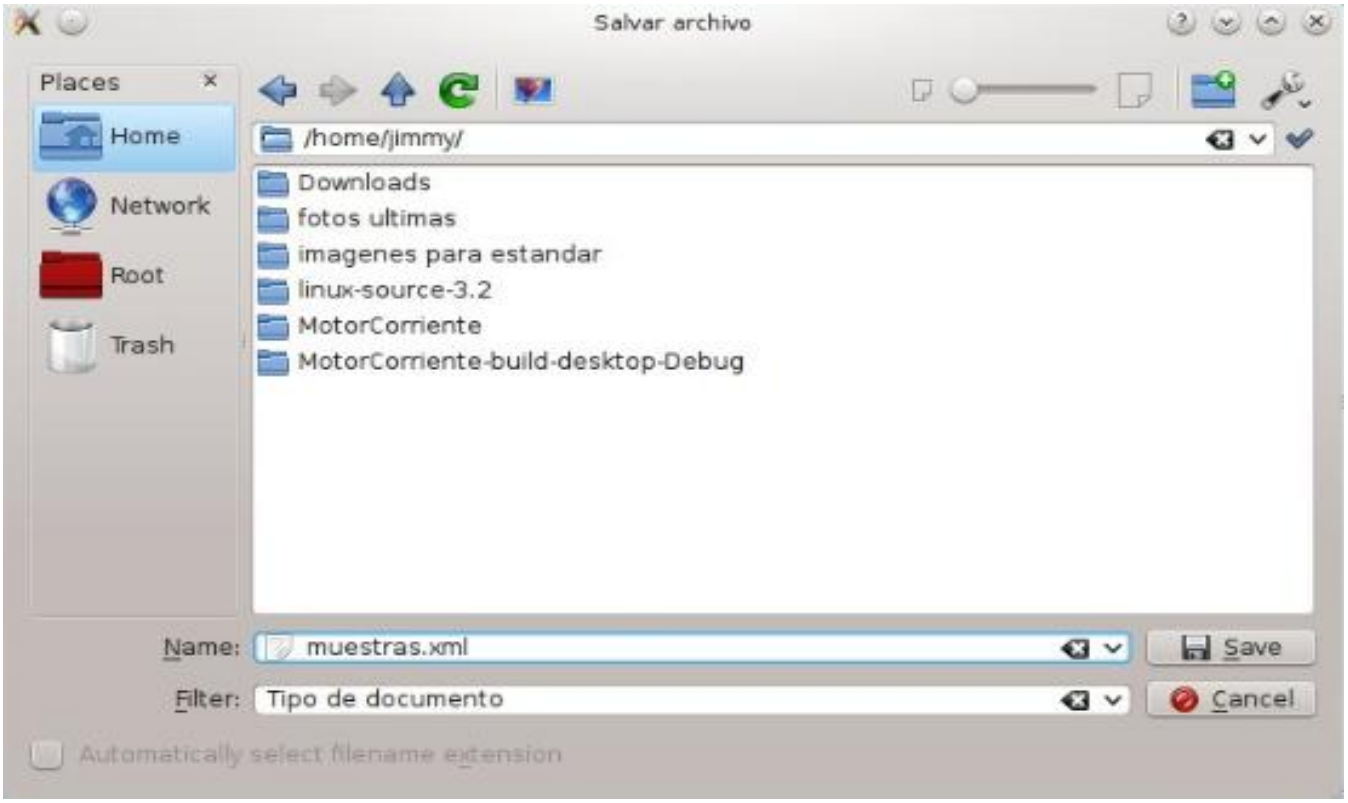
HU #6: Mostrar las muestras almacenadas.

Historia de Usuario	
Número: 6	**Nombre de Historia de Usuario:** Mostrar las muestras almacenadas.
Modificación de Historia de Usuario Número: -	
Usuario: Jimmy Bassa Martínez	**Iteración Asignada:** 3
Prioridad en negocio: Medio	**Puntos estimados**: 1
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1
Descripción: Visualiza en una tabla las muestras que se encuentran almacenadas en tiempo de ejecución.	
Observaciones:	
Prototipo de interfaz: 	

HU #7: Guardar muestras en un fichero.

Historia de Usuario	
Número: 7	**Nombre de Historia de Usuario:** Guardar muestras en un fichero.
Modificación de Historia de Usuario Número: -	
Usuario: Jimmy Bassa Martínez	**Iteración Asignada:** 3
Prioridad en negocio: Medio	**Puntos estimados**: 1
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1
Descripción: Permite guardar en un fichero con extensión ".xml" las muestras almacenadas.	
Observaciones:	

Prototipo de interfaz:



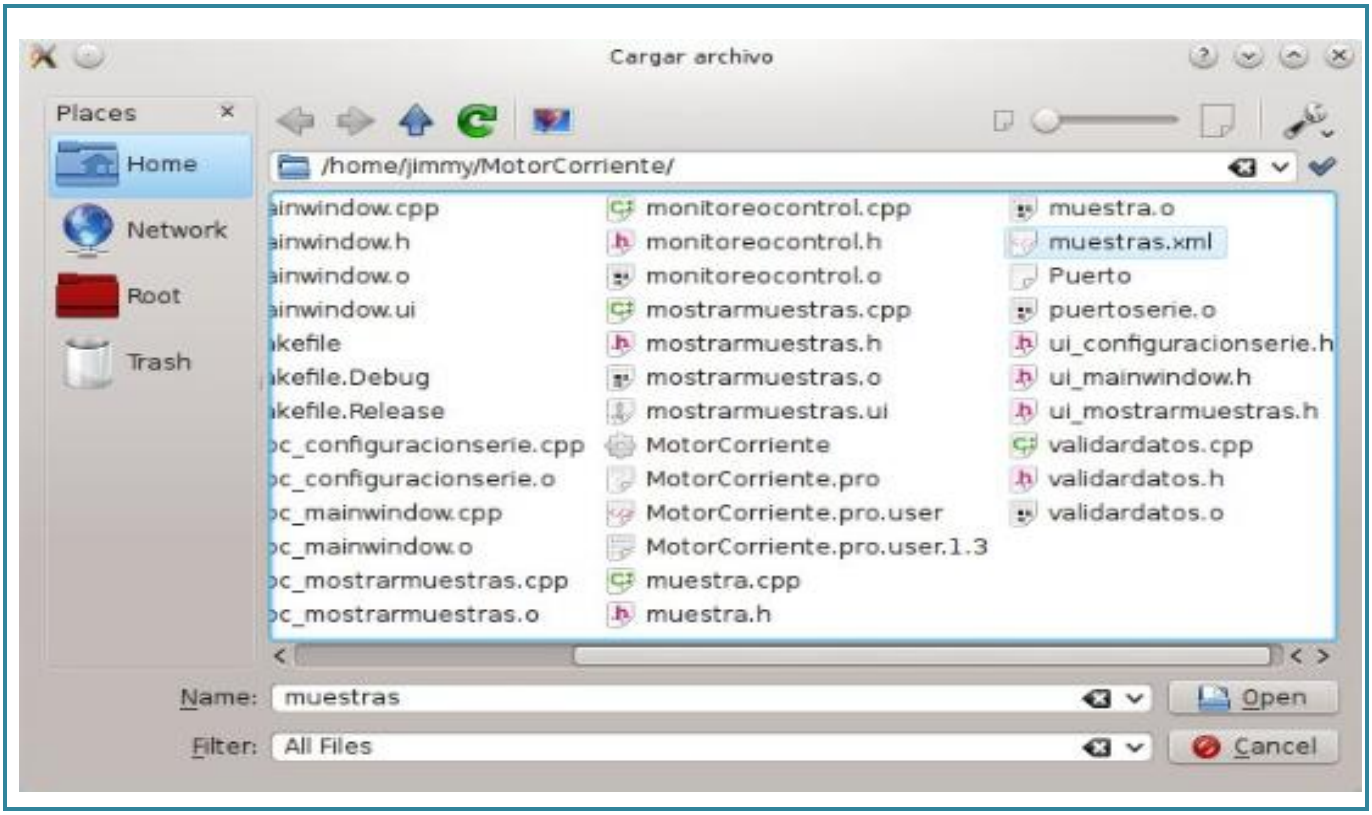
HU #8: Graficar muestras almacenadas.

Historia de Usuario	
Número: 8	**Nombre de Historia de Usuario:** Graficar muestras almacenadas.
Modificación de Historia de Usuario Número: -	
Usuario: Jimmy Bassa Martínez	**Iteración Asignada:** 3
Prioridad en negocio: Medio	**Puntos estimados**: 1

Riesgo en Desarrollo: Medio	**Puntos Reales:** 1
Descripción: Permite al usuario escoger una muestra almacenada para realizar su graficación.	
Observaciones:	
Prototipo de interfaz:	

HU #9: Cargar muestras desde fichero.

Historia de Usuario	
Número: 9	**Nombre de Historia de Usuario:** Cargar muestras desde fichero.
Modificación de Historia de Usuario Número: -	
Usuario: Jimmy Bassa Martínez	**Iteración Asignada:** 3
Prioridad en negocio: Medio	**Puntos estimados**: 1
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1
Descripción: Permite cargar las muestras desde un fichero con extensión ".xml".	
Observaciones:	
Prototipo de interfaz:	



HU #11: Mostrar mensaje de alarma.

Historia de Usuario	
Número: 11	**Nombre de Historia de Usuario:** Mostrar mensaje de alarma.
Modificación de Historia de Usuario Número: -	
Usuario: Jimmy Bassa Martínez	**Iteración Asignada:** 4
Prioridad en negocio: Bajo	**Puntos estimados**: 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1

Descripción: Se muestra un mensaje de alarma si en el momento de la lectura del parámetro se obtiene un valor no permitido.
Observaciones:
Prototipo de interfaz:

Anexo II – Tarjetas CRC

Tarjeta CRC. Clase ConexionSerie.

Tarjeta CRC	
Datos de la clase	
Nombre de la clase: ConexionSerie	
Descripción: Es la encargada de establecer la conexión con el puerto serie, además de enviar y recibir datos por mediación de este.	
Responsabilidades	**Colaboradores**
abrirPuerto cerrarPuerto enviarMensaje recibirMensaje getConectado	QextSerialPort

Tarjeta CRC. Clase Muestra.

Tarjeta CRC	
Datos de la clase	
Nombre de la clase: Muestra	
Descripción: Es la encargada de almacenar los valores de las variables del motor.	
Responsabilidades	**Colaboradores**
salvarXML cargarXML getPulsaciones getVelocidad getTiempos getCantReal getFecha getHora media varianza	QXmlStreamReader QXmlStreamWriter

Tarjeta CRC. Clase VentanaPrincipal.

Tarjeta CRC

Datos de la clase	
Nombre de la clase: VentanaPrincipal	
Descripción: Permite modificar y leer los parámetros del motor, y muestra gráficamente la tendencia de los parámetros del motor en el tiempo.	
Responsabilidades	**Colaboradores**
iniciarLectura graficar iniciarSimulacion detenerSimulacion restaurarValores regularPulsaciones mostrarMuestras validarValores salvarMuestras cargarMuestras	MonitoreoControl QwtPlotCurve QwtPlotGrid

Tarjeta CRC. ConfiguracionSerie.

Tarjeta CRC
Datos de la clase

Nombre de la clase: ConfiguracionSerie	
Descripción:	
Responsabilidades	**Colaboradores**
conectarse validarDatos	ValidarDatos MonitoreoControl

Tarjeta CRC. ValidarDatos

Tarjeta CRC	
Datos de la clase	
Nombre de la clase: ValidarDatos	
Descripción:	
Responsabilidades	**Colaboradores**
texto real entero	QString

Anexo III – Tareas de Ingeniería

Tarea de Ingeniería # 6 Modificar el valor de las pulsaciones del motor.

Tarea de Ingeniería	
Número Tarea: 6	**Número Historia de Usuario:** 4
Nombre Tarea: Modificar el valor de las pulsaciones del motor.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3 días
Fecha Inicio: 17/3/2014	**Fecha Fin:** 20/3/2014
Programador Responsable: Jimmy Bassa Martínez	
Descripción: Permite regular las pulsaciones del motor.	

Tarea de Ingeniería # 7. Reiniciar sistema.

Tarea de Ingeniería	
Número Tarea: 7	**Número Historia de Usuario:** 5
Nombre Tarea: Reiniciar sistema.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3 días
Fecha Inicio: 21/3/2014	**Fecha Fin:** 24/3/2014
Programador Responsable: Jimmy Bassa Martínez	
Descripción: Permite al usuario atreves de la barra de menú ejecutar la operación Reiniciar Sistema para poder realizar otras operaciones en la aplicación.	

Tarea de Ingeniería # 8. Visualizar las muestras almacenadas.

Tarea de Ingeniería	
Número Tarea: 8	**Número Historia de Usuario:** 6
Nombre Tarea: Visualizar las muestras almacenadas.	

Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3 días
Fecha Inicio: 27/3/2014	**Fecha Fin:** 30/3/2014
Programador Responsable: Jimmy Bassa Martínez	
Descripción: Para realizar esta operación el usuario deberá ejecutar la operación Mostrar muestras que se encuentra en la barra de menú, y estas se visualizarán en una tabla en caso de que se encuentren almacenadas.	

Tarea de Ingeniería # 9. Guardar muestras en un fichero.

Tarea de Ingeniería	
Número Tarea: 9	**Número Historia de Usuario:** 7
Nombre Tarea: Guardar muestras en un fichero.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3 días
Fecha Inicio: 1/4/2014	**Fecha Fin:** 4/4/2014
Programador Responsable: Jimmy Bassa Martínez	
Descripción: Para realizar esta operación el usuario deberá ejecutar la operación Salvar muestra que se encuentra en la barra de menú, y estas se guardarán en un archivo “.xml”.	

Tarea de Ingeniería # 10. Graficar muestras almacenadas.

Tarea de Ingeniería	
Número Tarea: 10	**Número Historia de Usuario:** 8
Nombre Tarea: Graficar muestras almacenadas.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3 días
Fecha Inicio: 9/4/2014	**Fecha Fin:** 13/4/2014
Programador Responsable: Jimmy Bassa Martínez	

Descripción: Para realizar esta operación el usuario deberá tener almacenadas las muestras, y una vez podrá graficar las misma dando click derecho encima de la misma y esta será graficada con sus valores respectivamente.

Tarea de Ingeniería # 11. Cargar muestras.

Tarea de Ingeniería	
Número Tarea: 11	**Número Historia de Usuario:** 9
Nombre Tarea: Cargar muestras.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3 días
Fecha Inicio: 15/4/2014	**Fecha Fin:** 18/4/2014
Programador Responsable: Jimmy Bassa Martínez	
Descripción: Para realizar esta operación el usuario deberá tener almacenadas las muestras en un fichero con extensión ".xml", y una vez podrá cargar la misma.	

Tarea de Ingeniería # 12. Simular una muestra.

Tarea de Ingeniería	
Número Tarea: 12	**Número Historia de Usuario:** 10
Nombre Tarea: Simular una muestra.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3 días
Fecha Inicio: 20/4/2014	**Fecha Fin:** 23/4/2014
Programador Responsable: Jimmy Bassa Martínez	
Descripción: Permitirá al usuario simular la lectura de las variables que este introduzca al sistema manualmente.	

Tarea de Ingeniería # 13. Graficar simulación.

Tarea de Ingeniería	
Número Tarea: 13	**Número Historia de Usuario:** 10
Nombre Tarea: Graficar simulación.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3 días
Fecha Inicio: 24/4/2014	**Fecha Fin:** 27/4/2014
Programador Responsable: Jimmy Bassa Martínez	
Descripción: Permitirá al usuario graficar una simulación de los valores de las variables que este introduzca al sistema manualmente.	

Tarea de Ingeniería # 14. Mostrar mensaje de alarma.

Tarea de Ingeniería	
Número Tarea: 14	**Número Historia de Usuario:** 11
Nombre Tarea: Mostrar mensaje de alarma.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3 días
Fecha Inicio: 2/5/2014	**Fecha Fin:** 5/5/2014
Programador Responsable: Jimmy Bassa Martínez	
Descripción: Se muestran mensajes de alarmas en caso de leerse un valor fuera del rango establecido.	

Anexo IV – Caso de pruebas de aceptación.

Caso de prueba de aceptación: Graficar los valores de las variables del motor en tiempo real.

Caso de Prueba de Aceptación	
Código: HU3p1	HU3: Graficar los valores de las variables del

	motor en tiempo real.
Nombre: Graficar los valores de las variables del motor en tiempo real.	
Descripción: Se desea probar que el sistema permita graficar en la interfaz principal los valores de las variables (pulsaciones y velocidad) del motor en tiempo real.	
Condiciones de ejecución: Se deben estar leyendo los valores de las variables del motor en tiempo real.	
Entrada/ Pasos de ejecución: Se ejecuta al mismo tiempo en que se están leyenda los valores de las variables del motor.	
Resultado esperado: Se muestran las gráficas en la interfaz principal con los valores de las variables en el tiempo.	
Evaluación de la prueba: Aceptada.	

Caso de prueba de aceptación: Modificar valores del motor.

Caso de Prueba de Aceptación	
Código: HU4p1	HU4: Modificar el valor de las pulsaciones del motor.
Nombre: Modificar valores del motor.	
Descripción: Se desea probar que el sistema permite modificar el valor de la variable pulsación, provocando un cambio en la velocidad del motor.	
Condiciones de ejecución: Existir conexión serie con el motor.	
Entrada/ Pasos de ejecución: Modificar el valor de las pulsaciones mediante el regulador.	
Resultado esperado: El valor de las pulsaciones se modifica en el motor y provoca una variación en la velocidad del mismo.	

Evaluación de la prueba: Aceptada.

Caso de prueba de aceptación: Reiniciar Sistema.

Caso de Prueba de Aceptación	
Código: HU5p1	HU5: Reiniciar Sistema.
Nombre: Reiniciar Sistema.	
Descripción: Una vez terminada la lectura de las variables de los valores del motor de corriente y la graficación, se desea que el sistema pueda restaurar los valores para una posterior ejecución.	
Condiciones de ejecución: Seleccionar la opción Reiniciar Sistema.	
Entrada/ Pasos de ejecución: Acceder a la opción Reiniciar Sistema que se encuentra en la barra de menú.	
Resultado esperado: Se establecen los valores por defectos en los diferentes componentes.	
Evaluación de la prueba: Aceptada.	

Caso de prueba de aceptación: Mostrar las muestras almacenadas.

Caso de Prueba de Aceptación	
Código: HU6p1	HU6: Mostrar las muestras almacenadas.
Nombre: Visualizar las muestras almacenadas.	
Descripción: Se desea probar que el sistema permite mostrar las muestras almacenadas en tiempo de ejecución en una tabla.	
Condiciones de ejecución: Escoger la opción Mostrar muestras	

Entrada/ Pasos de ejecución: Acceder a la opción Mostrar muestras que se encuentra en la barra de menú.
Resultado esperado: Que el sistema muestre una interfaz con todas las muestras almacenadas en una tabla.
Evaluación de la prueba: Aceptada.

Caso de prueba de aceptación: Guardar muestras.

Caso de Prueba de Aceptación	
Código: HU7p1	HU7: Guardar muestras en un fichero.
Nombre: Guardar muestras.	
Descripción: Se desea probar que el sistema permita guardar las muestras en un fichero xml.	
Condiciones de ejecución: Seleccionar la opción Guardar muestras.	
Entrada/ Pasos de ejecución: Acceder a la opción Guardar muestras que se encuentra en la barra de menú.	
Resultado esperado: Que las muestras sean guardadas en la dirección seleccionada en el formato xml.	
Evaluación de la prueba: Aceptada.	

Caso de prueba de aceptación: Graficar muestras almacenadas.

Caso de Prueba de Aceptación	
Código: HU8p1	HU8: Graficar muestras almacenadas.
Nombre: Graficar muestras almacenadas.	
Descripción: Se desea probar que el sistema permite la selección y graficado de una de las muestras almacenadas.	

Condiciones de ejecución: Debe existir al menos una muestra almacenada.
Entrada/ Pasos de ejecución: Acceder a la operación Mostrar muestras. Dar click derecho sobre la muestra que se desea graficar y dar escoger la opción graficar.
Resultado esperado: Se muestran las gráficas con los valores de las variables de la muestra seleccionada.
Evaluación de la prueba: Aceptada.

Caso de prueba de aceptación: Cargar muestras.

Caso de Prueba de Aceptación	
Código: HU9p1	HU9: Cargar muestras desde fichero.
Nombre: Cargar muestras.	
Descripción: Se desea probar que el sistema permita cargar las muestras desde un fichero xml.	
Condiciones de ejecución: Existir un fichero xml con muestras almacenadas.	
Entrada/ Pasos de ejecución: Acceder a la opción Cargar muestras, seleccionar el archivo a cargar.	
Resultado esperado: Se carga el archivo y las muestras cargadas pueden ser visualizadas.	
Evaluación de la prueba: Aceptada.	

Caso de prueba de aceptación: Simular una muestra.

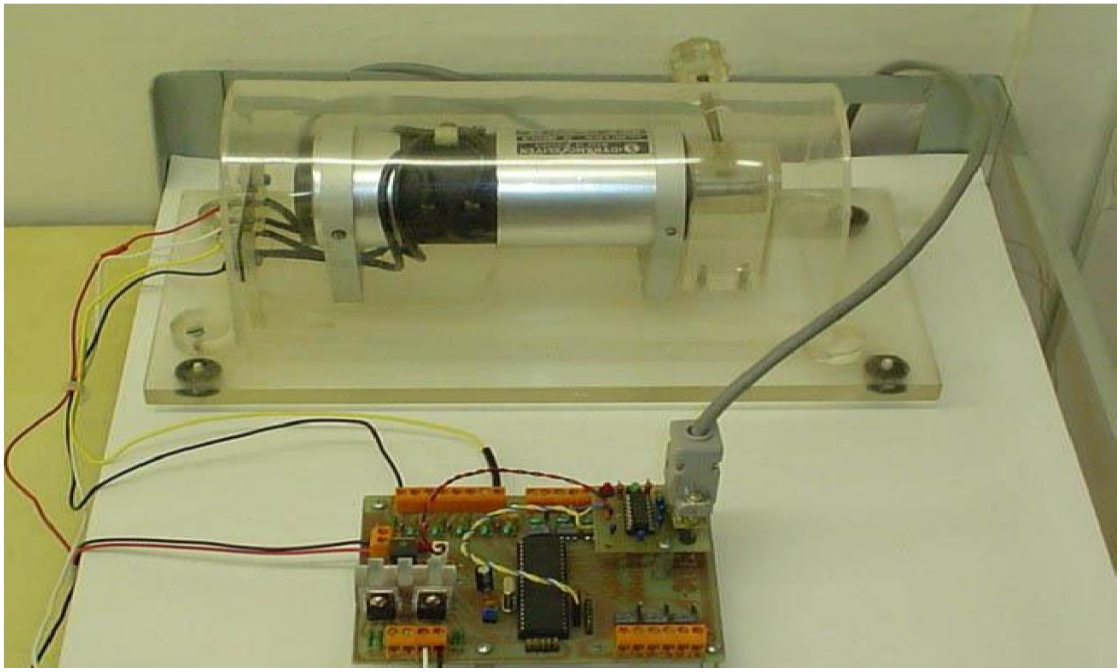
Caso de Prueba de Aceptación	
Código: HU10p1	HU10: Simular una muestra.

Nombre: Simular una muestra.
Descripción: Se desea que el sistema permita simular la lectura de los valores de las variables.
Condiciones de ejecución: Juego de datos correctos.
Entrada/ Pasos de ejecución: Insertar los valores de las variables y el intervalo de tiempo manualmente. Dar click en el botón simular que se encuentra en la interfaz principal.
Resultado esperado: Se muestran las gráficas con la simulación de los valores de las variables en el tiempo.
Evaluación de la prueba: Aceptada.

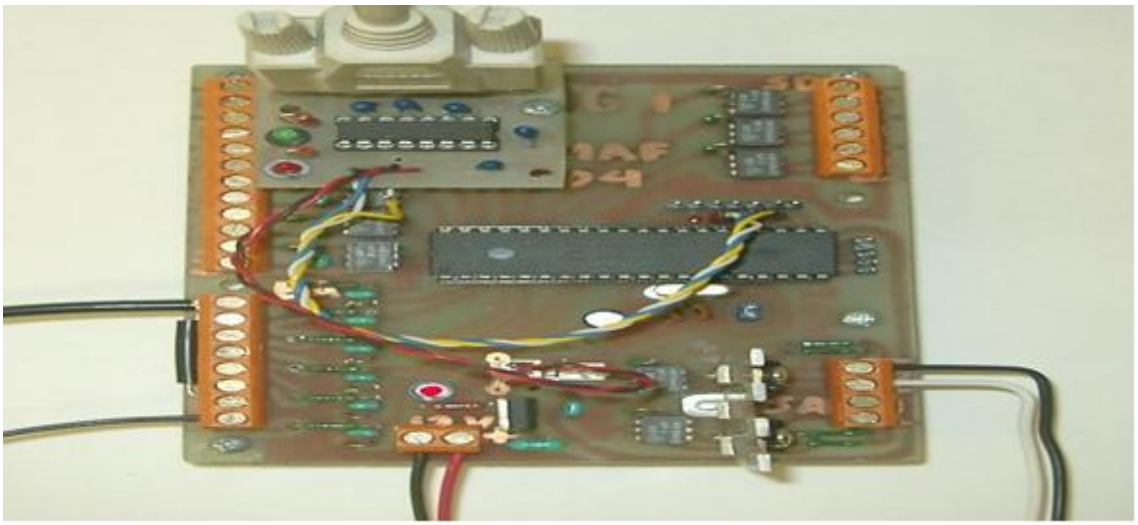
Caso de prueba de aceptación: Mostrar mensajes.

Caso de Prueba de Aceptación	
Código: HU11p1	HU11: Mostrar mensaje de alarma.
Nombre: Mostrar mensajes.	
Descripción: Se desea probar que el sistema muestre un mensaje de alarma en caso de la lectura de un valor fuera de rango.	
Condiciones de ejecución: Se deben estar leyendo los valores de las variables del motor.	
Entrada/ Pasos de ejecución: Se ejecuta al mismo tiempo en que se están leyenda los valores de las variables del motor.	
Resultado esperado: Se muestra una ventana con el mensaje de alarma.	
Evaluación de la prueba: Aceptada.	

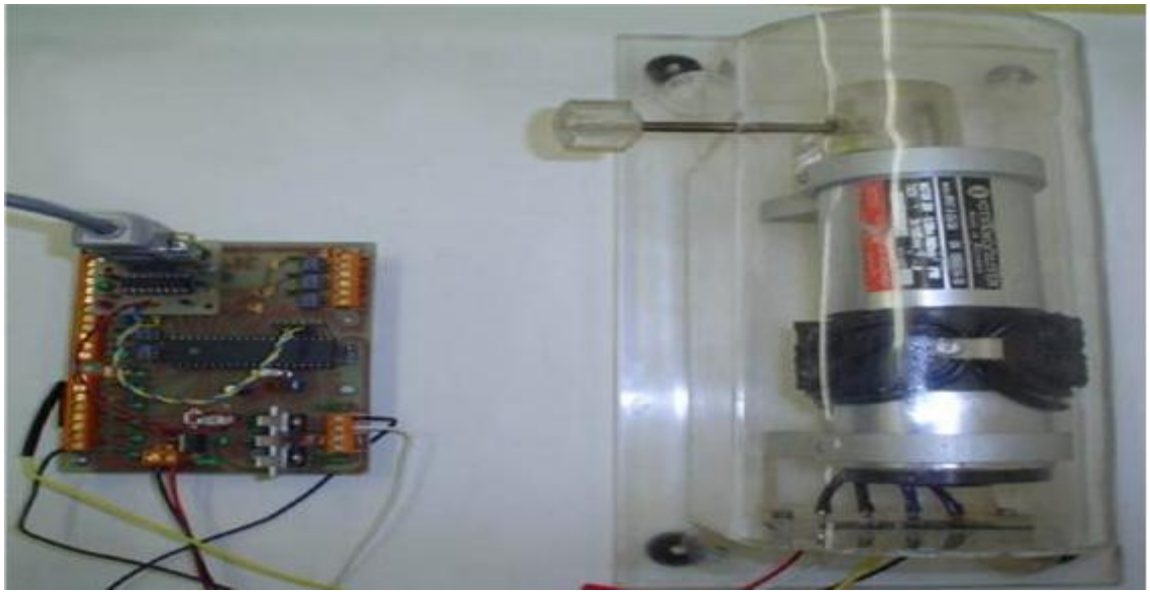
Anexo IV: Esquema y fotos de la maqueta del motor de CC conectado con la TDA.



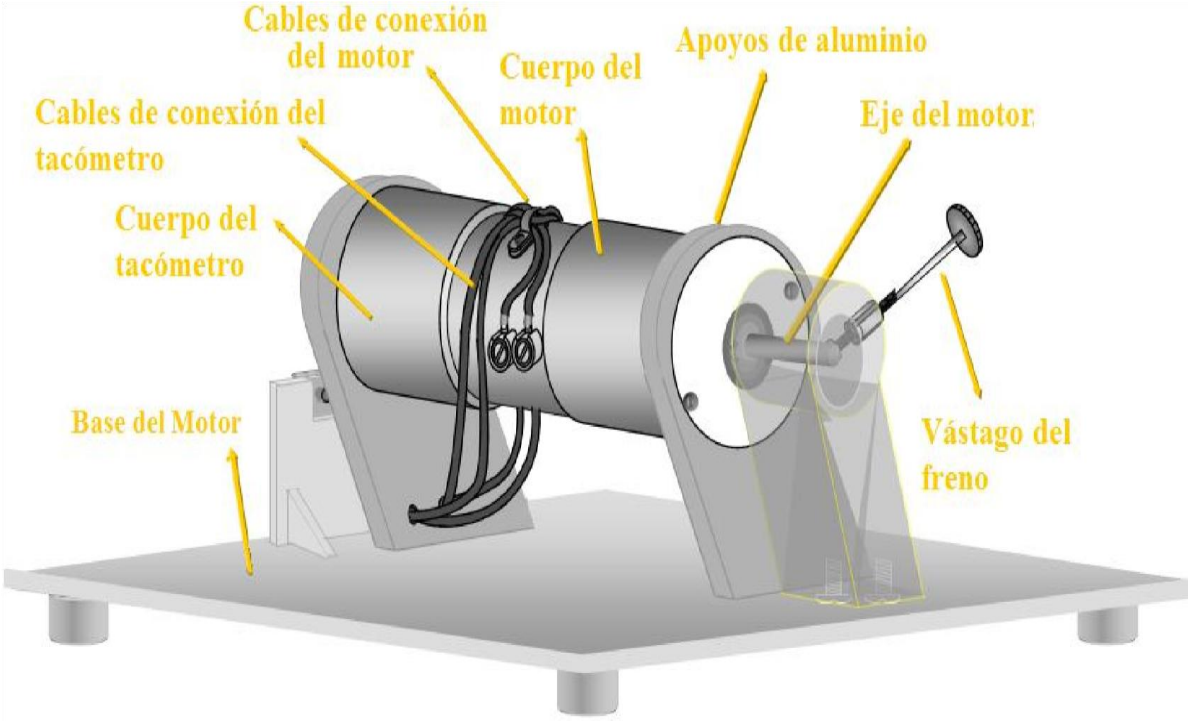
Maqueta del Motor de CC conectado con la TAD.



Vista superior de la TAD.



Vista superior del motor y la TAD.



Partes de la maqueta del motor.

Anexo V: Aval de aceptación.

ICIMAF

INSTITUTO DE CIEBERNÉTICA, MATEMÁTICA Y FÍSICA

MINISTERIO DE CIENCIA, TECNOLOGÍA Y MEDIO AMBIENTE

Aval de Desarrollo de Proyecto Externo

Por la presente a quien le pueda interesar, se hace constancia del desarrollo de la aplicación Sistema de monitoreo y control en tiempo real de un motor de corriente directa para prácticas de laboratorio docente. El sistema cumple con los requerimientos definidos que a continuación se mencionan:

1. Modificar parámetros de configuración del puerto serie.
2. Leer los valores de las variables del motor en tiempo real.
3. Graficar los valores de las variables del motor en tiempo real.
4. Modificar el valor de las pulsaciones del motor.
5. Reiniciar sistema.
6. Mostrar las muestras almacenadas.
7. Guardar muestras en un fichero.
8. Graficar muestras almacenadas.
9. Cargar muestras desde fichero.
10. Simular una muestra.
11. Mostrar mensaje de alarma.

Por tanto la tarea desarrollada se cataloga de satisfactoria.

Entregado: 1/07/2014

Al estudiante: Jimmy Bassa Martínez

Ing. Frank Rafael Pacareu de la Cruz:

Nombre y firma de Tutor ICIMAF

Dr. Armando Plasencia Salgueiro:

Nombre y firma Jefe de Departamento

ACADEMIA DE CIENCIAS DE CUBA
DEPTO. TEORIA DEL CONTROL
ICIMAF