Software para la interpretación de señales de Electrocardiograma orientado para estudiantes.  
Documento de arquitectura de software

Versión <1.0>

Historial de revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Descripción | Autor |
| <dd/mmm/yyyy> | <x.x> | <detalles> | <nombre> |
| 12-9-2015 | 1.0 | Documento de visión. | William Ramírez Ruiz  Jhon Jairo Castañeda. |
| 4-10-2015 | 1.0 | Especificación de requerimientos (E.R) | William Ramírez Ruiz  Jhon Jairo Castañeda. |
| 26-10-2015 | 1.1 | Correcciones documento E.R. | William Ramírez Ruiz  Jhon Jairo Castañeda. |
| 28-10-2015 | 1.0 | Especificación de casos de uso. | William Ramírez Ruiz  Jhon Jairo Castañeda. |
| 20-11-2015 | 1.0 | Documento de arquitectura de software | William Ramírez Ruiz  Jhon Jairo Castañeda. |

Tabla de contenido

1. Introducción 3

1.1 Propósito 3

1.2 Alcance 3

1.3 Definiciones, Acrónimos, y Abreviaciones 3

1.4 Referencias 3

1.5 Resumen 3

2. Representación arquitectónica 3

3. Metas de arquitectura y restricciones 3

4. Vista de casos de uso 3

4.1 Realización de casos de uso 3

5. Vista lógica 3

5.1 Resumen 3

5.2 Paquetes de diseño arquitectónicamente significantes 3

6. Vista de procesos 3

7. Vista de despliegue 3

8. Vista de implementación 3

8.1 Resumen 3

8.2 Capas 3

9. Vista de datos (opcional) 3

10. Tamaño y desempeño 3

11. Atributos de calidad 3

Documento de arquitectura de software

# Introducción

A la hora de comenzar a construir un software se debe tener en consideración algunos aspectos tales como su arquitectura. En este se definen ciertos patrones tales como el estilo arquitectónico a usar, el patrón de diseño y los idiomas a usar. El definir cada uno de los aspectos anteriormente mencionados a la hora de establecer la arquitectura de nuestro software será crucial, ya que en cierta medida estos proporcionara orden u agilidad al proceso de programación y diseño y facilitara el resolver un problema a la hora en que sea necesario.

Desde un enfoque general tendremos que cosas tales como definir un leguaje de programación u patrón de diseño facilitara el reusó de código y la corrección de este en el momento en que se manifieste un error durante una prueba, ya que estos al exigir ser corregidos puede llegar a manos de otro programador y este deberá seguir una lógica ya definida en la gestión del proyecto.

## Propósito

Este documento se centrara en un estilo arquitectónico basado en el uso de repositorios. Básicamente el usuario final tendrá que acceder a un repositorio en especifico en donde se aloja el software por lo cual este tendrá que tener inicialmente una conexión a internet para poder acceder al alojamiento en donde se encuentra este. Se implementara de esta manera con el propósito de brindara acceso a quien necesite de este.

Los repositorios son una gran herramienta debido a que en estos se pueden ir mejorando continuamente el software mediante la implementación de versiones, de tal manera se tendrá una plataforma en donde el usuario podrá tener a su alcance las futuras mejoras que se le haga a este software.

## Alcance

Debido a la arquitectura que se usara el alcance de esta estará en un gran segmento, se tendrá entonces que cualquier persona que manifieste interés en el uso y análisis de este tendrá a su disposición las estructuras usadas para el desarrollo de este, observando la lógica de cada versión y la mejoría y cambios que tiene este al avanzar de versión.

Aparte de ello si se manifiesta la necesidad en los repositorios se incluirán enlaces en donde se explicara más detalladamente como es el uso de cada clase usada y como es la funcionalidad de estas.

## Definiciones, Acrónimos, y Abreviaciones

Repositorio: Almacén o lugar donde se guardan ciertas cosas. En este contexto infiere a un lugar en donde reposaran los archivos correspondientes al software (librerías, versiones del software , etc).

## Referencias

Se tomaron como referencia los documentos ubicados en las siguientes direcciones:

* <https://s3.amazonaws.com/piazza-resources/idenwhs5uyd3ds/igvh7cuwhgv2u7/Ch6.pdf?AWSAccessKeyId=AKIAJKOQYKAYOBKKVTKQ&Expires=1448089867&Signature=MYgXrvDNkaSjDfKnNq74yP5i%2BRo%3D>
* <http://materias.fi.uba.ar/7510/practica/zips/DocArquitecturaEj2.pdf>

## Resumen

Para este software se usara una estructura basada en repositorios. Lo referente a los patrones de diseño este se orientara a un modelo de control centralizado en donde tendremos una división por sistemas o subsistemas. Como todo sistema de control tendremos que las decisiones recaerán prácticamente en las variables de estado, las cuales determinaran un curso para el control. Al tener una división entre sistemas y subsistemas se tiene que el sistema llamara un retorno a los subsistemas según las variables de estados proporcionadas en este.

# Representación arquitectónica

El software tendrá una orientación relacionada a un modelo de control centralizado, el cual se construirá a través de uso de librerías. Se tendrá entonces una parte principal que definiremos como el sistema el cual proporcionara las variables de estados para los subsistemas. Este sistema prácticamente será una interfaz en la cual el usuario al hacer un click generara una variable de estado, por ejemplo al presionar un botón generara un variable que arranque la comunicación serial u al seleccionar una opción en el menú desplegable genere una variable que inicie la clase grafica.

Tendremos entonces que el sistema será una interfaz compuesta por dos menús desplegables, dos botones inicio y parada y una cuadro de texto. Lo referente a los subsistemas podremos hacer una relación con las clases las cuales representaran los subsistemas, y a estos llegaran las variables de estado que llamaran la ejecución de esta. Con lo anterior tendremos que para este software tendremos algunas clases para la graficacion y otras para la importación de datos tanto seriales como de información visual.

# Metas de arquitectura y restricciones

Al adoptar este tipo de arquitectura tendremos algunas ventajas desde la perspectiva del estilo y el patrón utilizado. Al usar el estilo orientado a repositorios basado en el uso de capas, se tiene como meta proporcionar a un gran segmento de población la opción de usar este software según la necesidad. También al usar los repositorios se podrá suministrar una actualización continua al software mediante la creación de versiones permitiendo a aquellos interesados evidenciar el avance entre cada versión y la oportunidad de que ellos participen este proceso de mejoramiento del software por su cuenta.

Con respecto al patrón se hará uso de la metodología de control centralizado. Desde ciertas perspectivas tendremos una ventaja, esta será la fácil compresión de funcionamiento de software para el usuario, ya que se enfatizara en dos conceptos, sistema y subsistema. Como desventaja tendremos el hecho de que se crea una gran dependencia en el funcionamiento de este, delimitando su funcionamiento al sistema, ya que este generara las variables de estado necesario para la ejecución de los subsistemas.

# Vista de casos de uso

Lo referente a los casos de uso tendremos que desde cierta perspectiva la funcionalidad principal será lo referente a la comunicación serial de los datos. Debido a que el software se tornara como uno de adquisición de datos, la comunicación entre el microcontrolador y java será crucial, por lo cual la cobertura arquitectónica casi que se centrara principalmente en este proceso, rectificando antes que nada que la comunicación anteriormente descrita exista y que estos datos transmitidos sean los esperados.

## Realización de casos de uso

Para el funcionamiento del software tendremos que los casos de uso se manifestaran de la siguiente manera ante un escenario ideal. Inicialmente se inicial la comunicación serial y se verificara la correcta transmisión de estos, posterior a ello se almacenaran en un vector, tomando así una muestra de la señal. Posterior a ello al generarse esta variable en un vector otro caso de uso de encargara de tomar esos datos y graficarlos haciendo una relación amplitud vs tiempo y paralelamente en otro caso de uso este se tomara también y se operara con el fin de calcular la frecuencia cardiaca de la señal obtenida.

En otro apartado tendremos un caso de uso que permitirá la suministración de información mediante el uso de menús desplegables, estos al hacer accionados llamaran ciertas clases que contienen las graficas que suministraran esta información.

# Vista lógica

La arquitectura de este software tendrá una perspectiva orientada al desarrollo de clases las cuales tendrán cierta función plenamente establecida. La lógica se orientara entonces en la delegación de actividades por clases, en donde cada una ejecutara cierto proceso al generarse una variable de estado que llame a esta. Para el caso de este software se tendrá una delegación básica de tres funcionalidades, toma de datos, interpretación de datos y graficación de estos, y adicionalmente suministración de información visual para el usuario.

## Resumen

El software en cuestión de jerarquía tendrá un orden especifico debido a la dependencias que existen entre clases para la ejecución de procesos. Con ello se tendrá entonces en la parte superior una clase main la cual contendrá básicamente una interfaz que llamara a otras clases mediante el uso de “Actionlisteners”, posterior a ello tendremos la clase encargada de la comunicación serial, en donde se generaran las variables necesarias para ejecutar la clase graficacion. Como se puede observar hasta este punto la jerarquía tendrá gran importancia ya que si iniciar la interfaz no se puede arrancar la comunicación serial, sin la comunicación no se generara las variables y sin ellas no se podrá graficar. Además de esto se tendrá otras clases las cuales en si no tendrán un orden jerárquico entre ellas, aunque este podría tenerlo en cierta medida según la necesidad del usuario, puesto que estas clases meramente se encargaran de la suministración de información y la toma de este y el orden en que se tomen dependerá de la necesidad que tenga el usuario.

## Paquetes de diseño arquitectónicamente significantes

Desde la perspectiva arquitectónica tendremos que dos enfatizaciones. Por un lado tendremos un proceso que podría contenerse un paquete llamado comunicación, toma de datos y graficacion, las cuales tendrán como función básicamente comunicar, almacenar y graficar datos como su nombre lo indica. Este en nuestra arquitectura tendrá el mayor nivel de significancia.

Por otro lado se tendrá un paquete que denominaremos información, este tendrá como función la “graficacion” de la información en pantalla al momento de interactuar con un menú desplegable.

**Ver diagrama “paquetes de diseño”.**

# Vista de procesos

Como se menciono anteriormente existe gran dependencia a la hora de la ejecución de procesos. Se tendrá de entrada de la interfaz que denominaremos como clase main se crearan múltiples dependencias para la ejecución de procesos. Como se menciono en apartados anteriores la interfaz nos generara unas variables de estado, en donde estas prácticamente permitirán la ejecución de cada proceso al momento de interactuar con la interfaz.

La comunicación que se tendrá en el software desde la perspectiva de procesos será básicamente originada desde una clase main, esta ira a otra clase como parámetro de inicio, devolviendo el proceso que contiene dicha clase. Por lo cual tendremos que al interactuar con un botón “inicio” este nos arrojara una variables de estado en un Actionlistener el cual a su vez llamara a una clase comunicación y devolverá el proceso correspondiente a este. Para el caso del suministro de información en pantalla será algo con la misma lógica, a través de Actionlisteners se llamaran dichos procesos que devuelven las graficas de información que aparecerán en pantalla.

# Vista de despliegue

En el caso de las comunicaciones como tal del software con el hardware, se tendrá dos tipos de lógicas. Debido a que se usara un microcontrolador como tarjeta de adquisición se deberá crear comunicaciones de tipo serial debido a que un proceso requiere de la funcionalidad de adquirir datos de dicha tarjeta.

Con respecto a la cuestión de aspectos tales como suministración de información para los posibles usuarios, se tendrá una comunicación cliente-servidor, en donde el cliente será el programador y el servidor será el repositorio en donde se alojara la información correspondiente al software, este tendrá como ventaja el fácil mejoramiento del software mediante la creación de versión, ya que al alojarse en un servidor el programador solamente deberá encargarse de mantener versiones funcionales en este y el usuario final únicamente deberá acceder a este para descargar nuevas versiones u parches relacionados con correcciones, sin necesidad de que el programador u persona que ofrezca el software tenga que intervenir directamente en la maquina del usuario final.

# Vista de implementación

Al momento de incursionar en la arquitectura del software se deberá considerar antes que nada la creación de una línea base la cual deberá estar en un repositorio y a partir de esta comenzar con el mejoramiento continuo del software según se manifieste la necesidad , esto permitirá en cierta medida que el software mantenga un nivel de desempeño (manteniendo siempre una línea base y versiones funcionales), seguridad (proporcionando base de datos en donde se encuentre versiones funcionales en caso de que se presente algún fallo) y mantenimiento.

Por lo general al momento de hablar de arquitectura de software se crea una relación con el uso de capas, en este caso no será la excepción, el uso de capas será crucial para estructurar la lógica que llevara nuestro software. Por lo tanto se tendrá una totalidad de 3 capas, acceso a datos, uso de las clases enfocadas a la suministración de información, uso de las clases relacionadas con las comunicaciones y una presentación final para el usuario la cual será la interfaz a usar.

## Resumen

Desde la perspectiva de capas el software tendrá en su totalidad 3 capas, comunicación cliente-servidor, lógica de clases y interfaz.

En la comunicación se hará inferencia a la forma en la cual se suministra el software a un usuario final en este caso se hará mediante una comunicación cliente-servidor con apoyo en los repositorios, en donde el programador a través de una interfaz posteara su software en una base de datos y de la misma manera a través de una interfaz el usuario final descargara en su efecto el software una versión o parche de ese.

En la lógica de clases se demarcara como será la lógica de funcionamiento del software, estableciéndose las respectivas dependencias entre procesos y los caminos que se pueden seguir al generar un variable de estado a través de la interfaz.

En la capa interfaz u presentación final se suministrara ya la vista como tal que tendrá a su disposición el usuario, denotando cosas como los botones, menús, cajas de texto y sus respectivas funcionalidades la cual consistiría en llamara o regresar una clase ya definida.

**Ver diagrama “capas”.**

## Capas

* **Capa presentación o interfaz**= **Subsistema (menús)** este contiene lo correspondiente al suministro de información (clase Grapichs). Subsistema(boton) este contiene el proceso de comunicación serial, al generarse la variable de estado genera una clase u proceso en el cual se establece una comunicación con el arduino y java de forma serial.
* **Capa lógica clases de usos=** Esta capa contendrá el orden lógico que seguirá el software, de tal manera que tendremos una relación entre el llamamiento de clases y un orden en el cual se llaman y ejecutan estas. Desde cierta perspectiva se han establecidos dos posibles rutas a gran escala las cuales generara el usuario al usar la interfaz y crear una variable de estado. De tal manera tendremos por una lado que al generar una acción con “inicio” se llamara una librería, se creara una comunicación usado dicha librería, se almacenara los datos recibidos en un vector generándose las variables y por último se trabajaran u operaran estas variables (Graficas, calculo de datos de interés).
* **Capa comunicación cliente-servidor:** En este caso asumiremos de entrara que tendremos como clientes aquellos que accederán al servidor (programador y usuario final ), desde la perspectiva servidor, el programador actuara como cliente a la hora de hacer uso de su base de datos (repositorio) para subir su software a este y “el usuario final del software” actuara de mismo modo como cliente a la hora de acceder a la base datos para bajara y hacer uso de dicho software.

**Ver diagrama “capas”.**

# Vista de datos (opcional)

# Tamaño y desempeño

Nuestra arquitectura desde la perspectiva del desempeño tendrá una limitante referente hasta donde lograra comunicarse los datos y la velocidad de transmisión de estos, esto se verá afectado por el microcontrolador y la forma en la cual java recibe los datos. Como se ha podido evidenciar se tendrá una limitante relacionada con el almacenamiento de los datos en un vector, ya que para este proceso se requerirá de un denominado tiempo de muestreo, en donde se establece hasta que tiempo se tomaron las muestras de la señal y la frecuencia en que se hizo esto, haciendo inferencia a tiempo como capacidad o rango total y frecuencia como sensibilidad u velocidad de transmisión de datos.

# Atributos de calidad

Al usar la arquitectura anteriormente descrita tendremos una gran ventaja con respecto a la calidad y facilidad de mejoramiento. Ya que no será necesario el acceder a cada máquina que contenga este software se podrá hacer un mejoramiento continuo mediante versiones de forma global, de tal manera que el usuario al desear una nueva versión solamente acceda este repositorio y logre instalar por si mismo dicha versión.

Adicional a ello, al usar el patrón concebido en el control centralizado, se lograra establecer un orden con respecto a la ejecución de procesos, teniendo así un orden a la hora de ejecutar alguna tarea y en caso de manifestarse un error, se podrá relacionar de forma más sencilla de que subsistema proviene el error, al verificar a donde llega la variable de estado generada en la interfaz.