



TECNOLOGICO DE MONTERREY

CAMPUS QUERÉTARO

RECTORÍA ZONA CENTRO

DIRECCIÓN DE INGENIERÍA Y ARQUITECTURA

DEPARTAMENTO DE SISTEMAS COMPUTACIONALES

DESCRIPCIÓN DE EVALUACIÓN DE ARQUITECTURA DE SOFTWARE

Morpholoid

DISEÑO DE ARQUITECTURA DE SOFTWARE

Autor: Alejandro Fernández Vilchis A00889204

Querétaro, México 27 de noviembre 2013.

Resumen

El sistema Morpholoid busca resolver los problemas conectividad para interacción de robots del tipo Bioloid utilizando tele presencia a partir de la detección de patrones humanos replicados a transformaciones específicas de los robots según su tipo.

RESUMEN.....	2
1. DEFINICIÓN DE ARQUITECTURA DE SOFTWARE	11
1.1. INTRODUCCIÓN.....	11
1.2. DESCRIPCIÓN FUNCIONAL DEL SISTEMA	13
1.3. OPERACIÓN DEL SISTEMA	14
1.3.1. Recuperación	15
1.3.2. Traducción.....	15
1.3.3. Selección.....	16
1.3.4. Transformación	16
1.3.5. Implementación	16
1.4. <i>Atributos de Calidad Significativos</i>	18
1.4.1. Funcionalidad	18
1.4.1.1. Interoperabilidad	18
1.4.2. Confiabilidad.....	18
1.4.2.1. Tolerancia a Fallas	18
1.4.3. Eficiencia.....	19
1.4.3.1. Utilización de recursos.....	19
1.4.4. Mantenibilidad	19
1.4.4.1. Acoplamiento	19
1.3.4.2. Modularidad	19
1.4.5. Portabilidad	19
1.4.5.1. Adaptabilidad	19
1.4.5.2. Reemplazabilidad	20
1.1. <i>Problemáticas identificadas o características del sistema críticos</i>	20
1.6. DESCRIPCIÓN GENERAL DE LA SOLUCIÓN PROPUESTA.....	21
1.7. <i>Tecnologías Utilizadas</i>	23
1.7.1. C++	23
1.7.2. Kinect	23
1.7.3. Bioloid	24
1.8. <i>Componentes de Software Integrados</i>	25
1.8.1. Visual Studio	25
1.8.2. NUI Library	25
1.8.3. Bioloid Library.....	25
1.9. <i>Descripción de la Arquitectura General</i>	26
1.9.1. Arquitectura de Software	26
Morpholoid:	26
Library:.....	27
Robot:	27
1.9.2. Arquitectura de la Aplicación	28
1.9.3. Diseño del Software	29

1.10.	<i>Patrones del Diseño</i>	31
1.10.1.	Descripción del Patrón	31
1.10.2.	Aplicación	31
1.10.3.	Diagrama de Clases del Patrón	32
1.10.4.	Diagrama de Secuencia del Patrón	32
1.10.5.	Descripción del Patrón	32
1.10.6.	Aplicación	33
1.10.7.	Diagrama de Clases del Patrón	33
1.10.8.	Diagrama de Secuencia del Patrón	34
1.10.9.	Descripción del Patrón	34
1.10.10.	Aplicación	34
1.10.11.	Diagrama de Clases del Patrón	35
1.10.12.	Diagrama de Secuencia del Patrón	35
2.	EVALUACIÓN DE LA ARQUITECTURA DE SOFTWARE	36
2.1.	<i>Técnica de Evaluación Utilizada</i>	36
	<i>Procedimiento de Evaluación</i>	36
	<i>Las fases del método son las siguientes:</i>	37
	<i>Árbol de Utilidad</i>	38
	<i>Diseño y Ejecución de Pruebas</i>	40
	<i>Prueba 1.1 Se hace un flujo total del proceso de capas en orden normal</i>	40
	Proceso	40
	Responsable de ejecución: Arquitecto	40
	Resultados Esperados	40
	Resultados Obtenidos	40
	<i>Prueba 1.2 Se hace un flujo normal haciendo cambios en el gestor de entrada (Kinect)</i>	41
	Proceso	41
	1.-Desconectar el Kinect de la computadora	41
	2.-Iniciar el programa	41
	Responsable de ejecución: Arquitecto	41
	Resultados Esperados	41
	El sistema deberá notificar al usuario que no hay un dispositivo Kinect conectado y deberá terminar correctamente	41
	Resultados Obtenidos	41
	<i>Prueba 1.3 Se hace un flujo normal haciendo cambios en el gestor de salida (Robot)</i>	42
	Proceso	42
	1.-Desconectar el robot de la computadora	42
	2.-Iniciar el programa	42
	3.-Seleccionar un tipo de robot	42
	4.-El usuario hace gestura de introducción de instrucción natural	42
	5.-El usuario hace gestura de instrucción para acción a Robot	42
	Responsable de ejecución: Arquitecto	42

Resultados Esperados.....	42
El sistema se ejecutará correctamente pero al no haber robot conectado no se verá un medio de salida. El sistema no deberá fallar al realizar la desconexión.....	42
Resultados Obtenidos	42
<i>Prueba 2.1 Se configura un robot diferente para un flujo normal del sistema</i>	<i>43</i>
Proceso	43
1.-Iniciar el programa.....	43
2.-Seleccionar un tipo de robot.....	43
3.-El usuario hace gestura de introducción de instrucción natural.....	43
4.-Desconectar el robot de la computadora.....	43
5.-El usuario hace gestura de instrucción para acción a Robot diferente al configurado.....	43
Responsable de ejecución: Relacionados	43
Resultados Esperados.....	43
Debido a que la acción de interacción física con el robot corre de manera paralela al sistema, el resultado de la ejecución al robot debe ser correcto aunque resulte en un movimiento diferente al esperado ya que no es el correcto, para el caso del sistema se espera que se ejecute correctamente.	
.....	43
Resultados Obtenidos	43
<i>Prueba 2.2 Se desconecta el robot durante un flujo normal del sistema</i>	<i>45</i>
Proceso	45
1.-Iniciar el programa.....	45
2.-Seleccionar un tipo de robot.....	45
3.-El usuario hace gestura de introducción de instrucción natural.....	45
4.-Desconectar el robot de la computadora.....	45
5.-El usuario hace gestura de instrucción para acción a Robot	45
Responsable de ejecución: Relacionados	45
Resultados Esperados.....	45
El sistema deberá soportar la desconexión del robot sin hacer cambios aparentes en el flujo del mismo.	45
Resultados Obtenidos	45
<i>Prueba 2.3 Se envía una instrucción que no pueda ser reconocida por el robot configurado</i>	<i>47</i>
Proceso	47
1.-Configurar una instrucción que no pueda ser realizada por el robot	47
3.-Seleccionar un tipo de robot.....	47
4.-El usuario hace gestura de introducción de instrucción natural.....	47
5.-Desconectar el robot de la computadora.....	47
6.-El usuario hace gestura de instrucción para acción a Robot.....	47
Responsable de ejecución: Arquitecto	47
Resultados Esperados.....	47

El sistema deberá soportar la instrucción mal configurada y dependiendo del robot o no ejecutará la carga de la instrucción o se ejecutará un movimiento diferente en el robot al normal, sin comprometer al sistema.	47
Resultados Obtenidos	47
<i>Prueba 2.4 Enviar valores inválidos para motores que no estén en modo llanta</i>	<i>48</i>
Proceso	48
1.-Configurar el robot móvil en el sistema y conectar físicamente el robot móvil.	48
2.-El usuario hace gestura de introducción de instrucción natural.....	48
3.-Desconectar el robot de la computadora.....	48
4.-El usuario hace gestura de instrucción para acción a Robot.....	48
Responsable de ejecución: Arquitecto	48
Resultados Esperados.....	48
El sistema no deberá introducir la instrucción al robot ya que los motores que no están en modo rueda no pueden tomar el tipo de instrucciones.	48
Resultados Obtenidos	48
<i>Prueba 2.5 Se envía una instrucción cuando no haya robot conectado.....</i>	<i>49</i>
<i>Proceso.....</i>	<i>49</i>
1.-Configurar el robot móvil en el sistema y desconectar físicamente cualquier robot.	49
2.-El usuario hace gestura de introducción de instrucción natural.....	49
3.-Desconectar el robot de la computadora.....	49
4.-El usuario hace gestura de instrucción para acción a Robot.....	49
Responsable de ejecución: Arquitecto	49
Resultados Esperados.....	49
El sistema deberá intentar escribir la instrucción sin éxito y notificar al usuario.	49
Resultados Obtenidos	49
<i>Prueba 3.1 Se ejecuta el proceso de capas sin lags de delay para realizar instrucciones en el robot.....</i>	<i>50</i>
<i>Proceso.....</i>	<i>50</i>
1.-Configurar el robot móvil en el sistema y desconectar físicamente cualquier robot.	50
2.-El usuario hace gestura de introducción de instrucción natural.....	50
3.-Desconectar el robot de la computadora.....	50
4.-El usuario hace gestura de instrucción para acción a Robot.....	50
Responsable de ejecución: Relacionados	50
Resultados Esperados.....	50
El sistema deberá ejecutar un ciclo de instrucciones continuo sin detenerse convirtiendo la gestura en instrucciones del robot de forma natural que el usuario pueda ver.	50
Resultados Obtenidos	50
<i>Prueba 3.1 Se ejecuta el proceso de capas con tiempo suficiente para ejecutar instrucciones por separado que sean distinguibles por el usuario.....</i>	<i>51</i>
<i>Proceso.....</i>	<i>51</i>
1.-Configurar el robot móvil en el sistema y desconectar físicamente cualquier robot.	51

2.-El usuario hace gestura de introducción de instrucción natural.....	51
3.-Desconectar el robot de la computadora.....	51
4.-El usuario hace gestura de instrucción para acción a Robot.....	51
Responsable de ejecución: Relacionados	51
Resultados Esperados.....	51
El sistema deberá mostrar aparentemente al usuario que ejecuta instrucciones pausadas sobre el mismo tipo de gestura dejando ver que se ejecuta correctamente.....	51
Resultados Obtenidos	51
<i>Prueba 4.1 Se evalúa la capa getter para las gesturas</i>	<i>52</i>
Proceso	52
1.-Segmentar el sistema a la capa getter.	52
Responsable de ejecución: Arquitecto	52
Resultados Esperados.....	52
Se espera que dependiendo del dispositivo de entrada configurado Kinect, se obtenga una gestura.	52
Resultados Obtenidos	52
<i>Prueba 4.2 Se evalúa la capa selection para la acción.....</i>	<i>53</i>
Proceso	53
1.-Segmentar el sistema a la capa selection.	53
Responsable de ejecución: Arquitecto	53
Resultados Esperados.....	53
Se espera que a partir de una gestura dada se obtenga una acción concreta.....	53
Resultados Obtenidos	53
<i>Prueba 4.3 Se evalúa la capa translation para la acción del robot</i>	<i>54</i>
Proceso	54
1.-Segmentar el sistema a la capa translation.....	54
Responsable de ejecución: Arquitecto	54
Resultados Esperados.....	54
Se espera que a partir de una acción dada se obtenga una acción concreta contenida en una configuración de robot.	54
Resultados Obtenidos	54
<i>Prueba 4.4 Se evalúa la capa transformation para las instrucciones del robot</i>	<i>55</i>
Proceso.....	55
1.-Segmentar el sistema a la capa transformation.	55
Responsable de ejecución: Arquitecto	55
Resultados Esperados.....	55
Se espera que a partir de una acción concreta del robot se genere un conjunto de instrucciones específicas del robot configurado.	55
Resultados Obtenidos	55
<i>Prueba 4.5 Se evalúa la capa de implementación para el Robot</i>	<i>56</i>
Proceso.....	56

1.-Segmentar el sistema a la capa implementación.	56
Responsable de ejecución: Arquitecto	56
Resultados Esperados.....	56
Se espera que a partir de un conjunto de instrucciones el robot ejecute las mismas de acuerdo a su librería de ejecución en puerto serial.	56
Resultados Obtenidos	56
<i>Prueba 5.1 Se agrega una capa evaluativa al inicio del proceso de capas</i>	<i>57</i>
<i>Proceso.....</i>	<i>57</i>
1.-Abrir el entorno de desarrollo del sistema.	57
Responsable de ejecución: Relacionados	57
Resultados Esperados.....	57
Se espera que un usuario pueda agregar una capa al sistema de manera intuitiva y sencilla enfocándose más en lo que realiza la capa que en agregarla como tal. Además se espera que la nueva capa se acople de manera natural al sistema mientras cumpla con las condiciones que propone el sistema.	57
Resultados Obtenidos	57
<i>Prueba 5.2 Se agrega una capa evaluativa al medio del proceso de capas</i>	<i>58</i>
<i>Proceso.....</i>	<i>58</i>
1.-Abrir el entorno de desarrollo del sistema.	58
Responsable de ejecución: Relacionados	58
Resultados Esperados.....	58
Se espera que un usuario pueda agregar una capa al sistema de manera intuitiva y sencilla enfocándose más en lo que realiza la capa que en agregarla como tal. Además se espera que la nueva capa se acople de manera natural al sistema mientras cumpla con las condiciones que propone el sistema.	58
Resultados Obtenidos	58
<i>Prueba 5.3 Se agrega una capa evaluativa al final del proceso de capas</i>	<i>59</i>
<i>Proceso.....</i>	<i>59</i>
1.-Abrir el entorno de desarrollo del sistema.	59
Responsable de ejecución: Relacionados	59
Resultados Esperados.....	59
Se espera que un usuario pueda agregar una capa al sistema de manera intuitiva y sencilla enfocándose más en lo que realiza la capa que en agregarla como tal. Además se espera que la nueva capa se acople de manera natural al sistema mientras cumpla con las condiciones que propone el sistema.	59
Resultados Obtenidos	59
<i>Prueba 5.4 Se agrega una sub capa evaluativa al inicio de un módulo del sistema</i>	<i>60</i>
<i>Proceso</i>	<i>60</i>
1.-Abrir el entorno de desarrollo del sistema.	60
Responsable de ejecución: Relacionados	60
Resultados Esperados.....	60

Se espera que el usuario pueda agregar una nueva gestura de manera intuitiva y de manera simple, siendo natural la agregación y que no se comprometa el sistema.	60
Resultados Obtenidos	60
<i>Prueba 5.5 Se agrega una sub capa evaluativa al medio de un módulo del sistema</i>	<i>61</i>
Proceso	61
1.-Abrir el entorno de desarrollo del sistema.	61
Responsable de ejecución: Relacionados	61
Resultados Esperados.....	61
Se espera que el usuario pueda agregar una nueva acción al sistema de manera intuitiva y natural, sin comprometer el sistema ni su funcionalidad.	61
Resultados Obtenidos	61
<i>Prueba 5.6 Se agrega una sub capa evaluativa al final de un módulo del sistema</i>	<i>62</i>
Proceso	62
1.-Abrir el entorno de desarrollo del sistema.	62
Responsable de ejecución: Relacionados	62
Resultados Esperados.....	62
Se espera que el usuario pueda agregar una nueva acción al robot en el sistema de manera intuitiva y natural, sin comprometer el sistema ni su funcionalidad.....	62
Resultados Obtenidos	62
<i>Prueba 5.7 Se agrega una rutina nueva para una acción dada en el sistema para algún tipo de robot.....</i>	<i>63</i>
Proceso	63
1.-Abrir el entorno de desarrollo del sistema.	63
Responsable de ejecución: Relacionados	63
Resultados Esperados.....	63
Se espera que el usuario pueda agregar tipo de robot en el sistema de manera intuitiva y natural, sin comprometer el sistema ni su funcionalidad	63
Resultados Obtenidos	63
<i>Prueba 5.8 Se modifica una rutina existente para una acción dada de un robot.....</i>	<i>64</i>
Proceso	64
1.-Abrir el entorno de desarrollo del sistema.	64
Responsable de ejecución: Relacionados	64
Resultados Esperados.....	64
Se espera que el usuario pueda agregar una acción a un tipo de robot en el sistema de manera intuitiva y natural, sin comprometer el sistema ni su funcionalidad.	64
Resultados Obtenidos	64
<i>Prueba 5.9 Se elimina una rutina existente de un robot para una acción dada.....</i>	<i>65</i>
Proceso	65
1.-Abrir el entorno de desarrollo del sistema.	65
Responsable de ejecución: Relacionados	65
Resultados Esperados.....	65

Se espera que el sistema soporte la eliminación repentina de una rutina sin comprometer la funcionalidad y estabilidad de todo el sistema.....	65
Resultados Obtenidos	65
<i>Conclusiones de las evaluaciones</i>	66
3. CONCLUSIONES GENERALES.....	67
4. BIBLIOGRAFÍA	68

1. Definición de Arquitectura de Software

1.1.Introducción

La robótica es una de las áreas de investigación con uno de los mayores índices de interés en las personas, tal vez sea por el mero hecho de que se ve como un avance científico, o tal vez sea porque es un tema controversial proveniente de la ciencia ficción donde las películas dejan volar a la imaginación en lo que los seres artificiales pueden lograr y como afectarán el futuro de la humanidad. Sin embargo existen muchos mitos acerca de la robótica y uno de ellos es la lejanía a la que se encuentra de las personas. La robótica está demasiado próxima y más que el inicio, en la actualidad está su apogeo, ya que en este momento ya existen aparatos robóticos que comienzan a ayudar en el día a día de la vida de la humanidad. Estos seres no son entes meramente pensantes, sino que alguien debió programarlos para que obtuvieran el grado de rasgos humanos que, desde la ciencia ficción, se han ido conociendo y se esperan cuando la revolución robótica tome lugar. Para unificar a la ciencia y a la robótica es necesario recuperar lo se conoce por ciencias básicas y adicionarle los beneficios de la tecnología en nuestro tiempo. Es así que existen muchas áreas y aplicaciones desde donde surge la robótica que es el gran tema de la Inteligencia Artificial, aquí la robótica es tan solo una pequeña rama del sin fin de teorías y aplicaciones que se han desarrollado tan solo en los últimos siglos.

Para poder interactuar con las personas los creadores de las máquinas que conocemos como robots han creado un gran número de formas de comunicarnos con ellos, desde la parte física que se convierten los sensores y la entrada de información que capta el robot, hasta la parte lógica que compone de algoritmos, procesamiento y entendimiento. Una de las últimas tecnologías que permite la interacción humano computadora es el dispositivo conocido como Kinect. Este dispositivo permite mediante la captura de imágenes y audio obtener la información de los movimientos y directivas de una persona para poder procesarlas en un sistema. Comúnmente este

dispositivo se ha utilizado en el área de videojuegos, sin embargo las personas han empezado a utilizarlo con fines experimentales para desarrollar sistemas y aplicaciones que permitan un cambio en la forma de interactuar con las computadoras. Desde este punto comienzan a ramificarse una serie de caminos sobre las áreas en las que se puede experimentar con el Kinect, como gráficas computacionales e inteligencia artificial, esta última tiene una gran rama de ciencias entre ellas la robótica.

Con el frente de colisión entre el Kinect y la Robótica se tienen diferentes publicaciones en el estado del arte para esta área específica del conocimiento. Siendo muy pronto para establecer un límite, ya que existe una gran variedad de aplicaciones en la robótica, como un gran número de robots cada una de las aplicaciones que utiliza ambas tecnologías se construye en sí misma ya sea por la arquitectura que utiliza, o por el lenguaje de programación al que está dirigido incluso por el tipo de instrucciones que se generan a partir de cada proyecto.

Una gama de robots que se utilizan para investigación mayormente en las universidades, son los robots de ensamblaje que se pueden modificar como si fueran rompecabezas en un gran número de robots diferentes. Tal es el caso del modelo Bioloid de la compañía Robotis, el cuál obtiene el nombre ya que se define en sí mismo como un robot capaz de obtener cualquier forma de un ser con características de vida en el planeta siendo desde: insectos, animales e incluso humanos.

Retomando la tecnología del Kinect y el último robot descrito, Bioloid, se encuentran una gran variedad de aplicaciones para ambos elementos como el uso del Kinect como control de posición de brazos para la ejecución de rutinas en robots móviles desarrollados con el Bioloid, o también el uso del Kinect como medio de replicación de los movimientos de un robot tipo humanoide, e incluso el uso del Kinect como medio de transformación del universo humano al universo entendido por un robot humanoide traslapando la física como medio de interacción, desarrollo y movimiento. Este tipo de aplicaciones en su

mayoría generan código desde el Kinect directamente hacia el robot sin utilizar un marco de desarrollo fijo como un framework o librería.

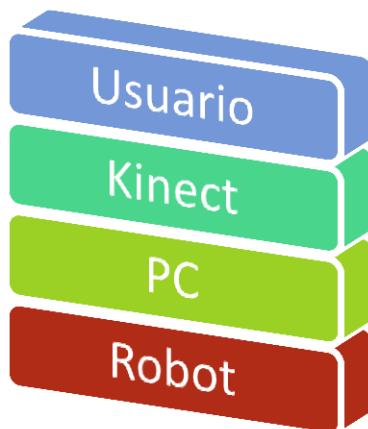
Uno de los principales problemas para el desarrollo de nuevos algoritmos complejos para la interacción robótica es la falta de sistemas bien definidos que permitan una mayor interacción con un gran número de robots diferentes.

Es por esta necesidad que se propone crear toda una arquitectura que permita acoplar de manera simple la interacción proveniente de datos de entrada como son el Kinect y los sensores del Bioloid para poder efectuar rutinas o movimientos a respuesta de la interacción según sea la morfología que tenga el robot para la ejecución de la aplicación en tiempo real.

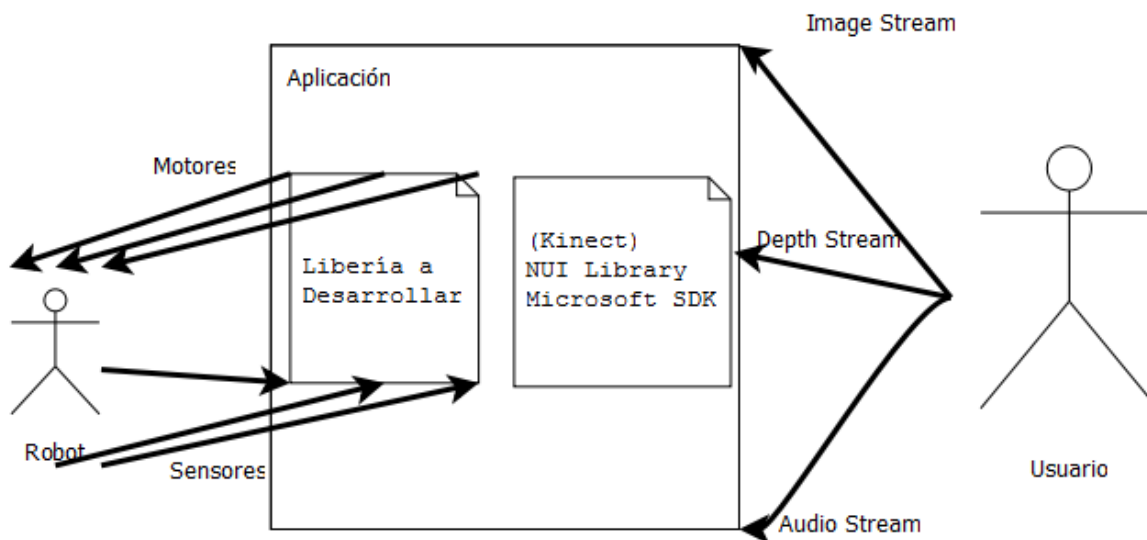
1.2.Descripción Funcional del Sistema

El usuario será capaz de introducir una instrucción a la aplicación mediante el Kinect y según el tipo de robot este responderá con la rutina necesaria para efectuar una interacción en el mundo real. Las instrucciones que pueden ser introducidas por el Kinect están definidas por la recopilación de los puntos que reconoce el esqueleto humano en los planos x,y,z. A partir de la información recopilada la aplicación formará un conjunto de instrucciones las cuales introducirá en el sistema arquitectónico desarrollado para que sean interpretadas por el tipo de robot que se está utilizando.

Las capas que utiliza la aplicación son las siguientes:

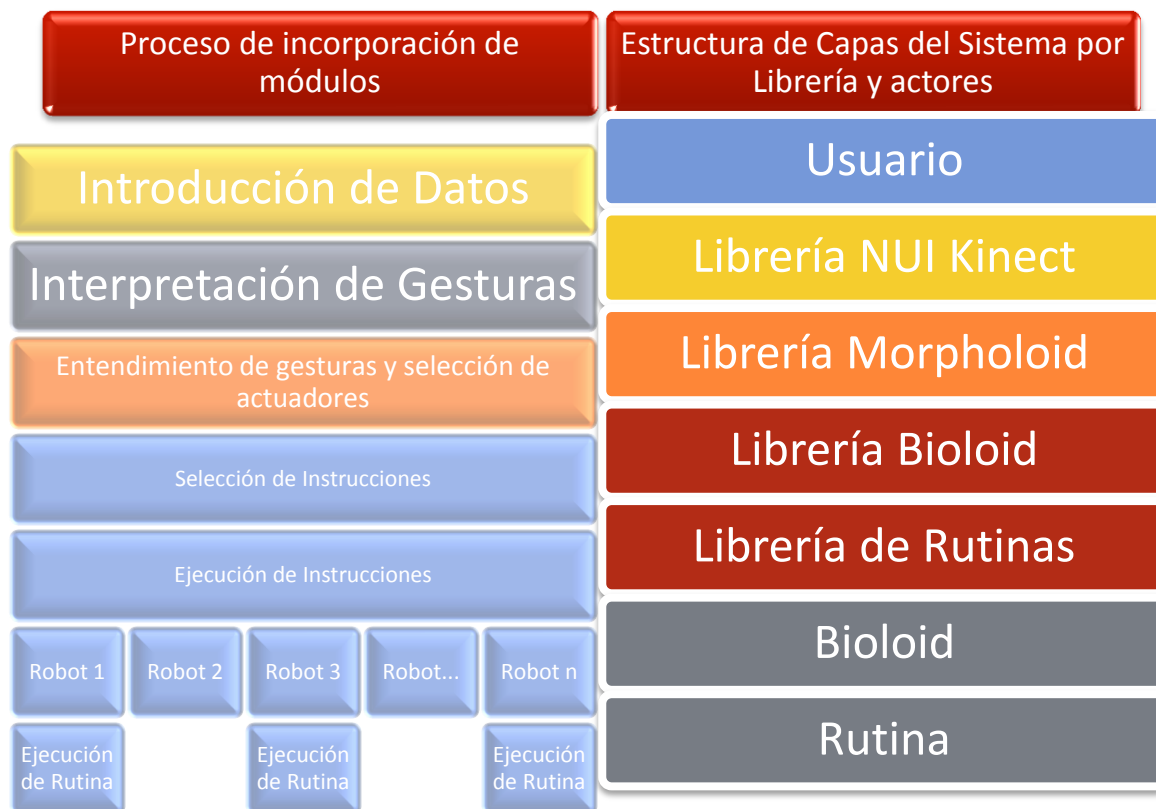


Para la estructura de interconectividad se tiene lo siguiente:



1.3. Operación del Sistema

El sistema está desarrollado por capas las cuales incorporan acciones que serán descritas más adelante. Las capas se conforman de la siguiente manera:



La manera en que funcionará el sistema se compone de varias capas que pueden definirse como: recuperación, traducción, generación, selección, transformación, implementación y ejecución. Se describen cada una de ellas:

1.3.1. Recuperación

Esta capa es intermedia a la interacción del usuario con un dispositivo de entrada, para este proyecto se utilizará el Kinect como medio de interpretación de gesturas del usuario, sin embargo el proyecto estará diseñado para poder interpretar cualquier tipo de gestura y extender el sistema a adicionar nuevas con diferentes dispositivos de entrada.

1.3.2. Traducción

La capa de traducción se encuentra alineada con la de recuperación ya que es la encargada de convertir una gestura en una acción dada de acuerdo a los patrones establecidos, igualmente que en la capa de recuperación el sistema será capaz de extender esta funcionalidad en

más acciones. Dependiendo del tipo de acción está podrá estar asociada a una o varias gesturas (1-1 o n-1), sin embargo estará limitado a no poder delimitar varias acciones hacia una gestura, esto para evitar que el sistema pueda llevar a inconsistencias o fallas en tiempo de ejecución debido a una mala interpretación de la capa de recuperación que pueda comprometer el paso hacia las siguientes capas.

1.3.3. Selección

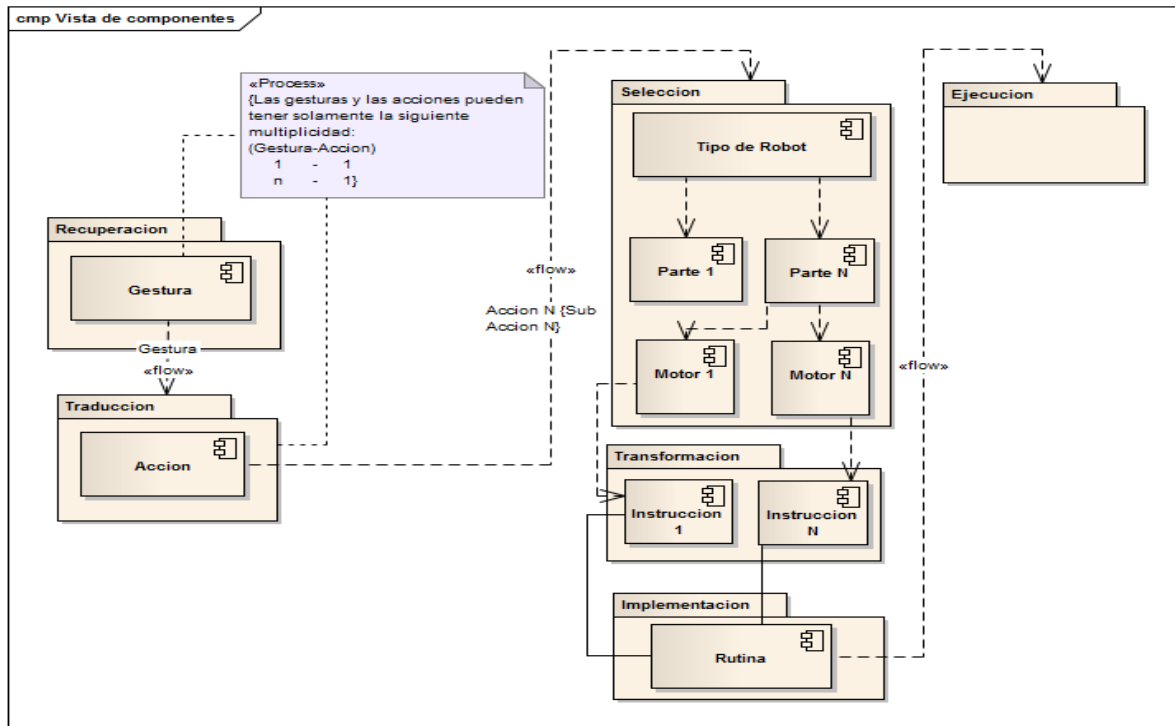
En la capa de selección se obtiene un conjunto específico de acciones las cuales van enfocadas a un autómata en especial. Por ejemplo: Si se tiene el brazo del robot y se desea moverlo, la acción de movimiento tiene que pasar a la capa de selección la cual indica que lo que se va a mover es el brazo de un tipo específico de robot Bioloid al cual le pertenecen un número definido de motores con un identificador específico. Una vez seleccionado todos los instrumentos requeridos para la acción se procede a pasar a la siguiente capa.

1.3.4. Transformación

La capa de transformación sirve para convertir una acción que proviene de la capa de selección en un conjunto finito de instrucciones que resuelvan la misma. Es en este punto que al resolverse varios conjuntos de sub acciones dentro de esta capa que entonces se generan las partes que resuelven una acción principal provista por la gestura de la capa de recuperación.

1.3.5. Implementación

En la capa de implementación se convierte un conjunto de instrucciones en forma que el robot pueda entenderlas de manera concreta, fluida y simple. Este paso es el último que se debe realizar y una vez introducidas las instrucciones a nivel del robot se puede ver físicamente en el robot una rutina dada como final del proceso del sistema.



1.4. Atributos de Calidad Significativos

Los atributos de calidad más significativos para el Sistema siguiendo el modelo ISO 9126 para Arquitecturas de Software son:

1.4.1. Funcionalidad

El Sistema debe cumplir completamente con su propósito y en caso de tener alguna falla como puede ser de interconectividad en las capas más externas deberá retroalimentar al usuario sobre cómo resolverlas o en su defecto manejarlas para que sean completamente transparentes.

1.4.1.1. Interoperabilidad

El sistema será capaz de manejar de manera efectiva la interconexión con todas sus capas así como en caso de modificar o cambiar alguna mantener los módulos de conectividad para hacer más sencillo la adaptación de una nueva capa a operar con el sistema o en su defecto a incrementar el número de capas del mismo.

1.4.2. Confiabilidad

1.4.2.1. Tolerancia a Fallas

El sistema será capaz de recuperarse en caso de tener fallas de comunicación entre cada una de las capas intermedias ya que son específicas sobre datos generados por el mismo sistema. En el caso de las capas más externas hacia el usuario y hacia el Bioloid, el sistema deberá notificar sobre errores en la comunicación y transferencia de datos al usuario para que se repitan o en su defecto reinicie el sistema. Los principales problemas que pueden surgir a partir de esto son lecturas de gesturas imprecisas o desconocidas que manden una rutina al Bioloid, o el escribir un valor al robot que no sea adecuado a los límites del tipo montado, así como también se puede tener fallas en la transmisión de rutinas debido al uso de comunicación serial.

1.4.3. Eficiencia

1.4.3.1. Utilización de recursos

Esta relación de componentes espacio tiempo se manejará en el sistema como interconectividad con el Kinect y transferencia de datos a muy bajo nivel siendo meras instrucciones del micro controlador del Bioloid, siendo el proceso rápido y de muy bajo procesamiento además de que el sistema no poseerá una interfaz gráfica como tal reduciendo el nivel de recursos utilizados tanto por el Kinect, como por la computadora para efectos de despliegue.

1.4.4. Mantenibilidad

1.4.4.1. Acoplamiento

El sistema de capas estará diseñado para permitir la interconectividad de las mismas para efectos de comunicación entre objetos entrantes y salientes de cada una de ellas.

1.3.4.2. Modularidad

El sistema de capas permitirá a todo el sistema cambiar o modificar una sin afectar los otros módulos del mismo permitiendo extensión y crecimiento así como capacidad para mantención del mismo. Lo más importante del uso de la modularidad del sistema será el mantener los objetos propios de entrada y salida que deberá mantener el mismo para permitir la comunicación y acoplamiento entre cada una de las mismas.

1.4.5. Portabilidad

1.4.5.1. Adaptabilidad

El sistema será adaptable dentro de cada una de sus capas para incorporar más elementos como pueden ser diferentes dispositivos de entrada, diferentes tipos de robots Bioloid, incluso diferentes tipos de robots. Para agregar diferentes tipos de robots será necesario

incorporar una capa de conectividad que maneje los diferentes tipos de conexión con robots como pueden ser comunicación serial, zigbee, bluetooth, Wi-Fi, etc.

1.4.5.2. Reemplazabilidad

Al estar montado en forma de capas el sistema permite reemplazar cada una de ellas y establecer nuevas o agregar más al modelo. Esto se hace en conjunto con la adaptabilidad para permitir mantención y extensibilidad del sistema hacia tipos diferentes de interacción con diferentes tipos de robots.

1.1. Problemáticas identificadas o características del sistema críticos

Entre las principales problemáticas que presenta el sistema, se destacan las relacionadas con los atributos de calidad como: adaptabilidad, modularidad, utilización de recursos y tolerancia a fallas.

- En primer lugar se tiene que para poder realizar el sistema de capas estas se deben comunicar efectivamente de acuerdo a los datos que entran y salen de las mismas, es por esto que al ser unario el tipo de entrada y salida se debe agrupar muy bien los datos de traspaso ya que de lo contrario se pueden enviar datos erróneos haciendo fallar en su totalidad al sistema sin tomar en cuenta las protecciones de tolerancia a fallos.
- En segundo lugar se tiene la utilización de recursos como uno de los principales problemas del sistema ya que el manejo de bajo nivel de datos no implica que no se haga procesamiento pesado dentro de la computadora ya que se procesará la imagen proveniente del Kinect en el esqueleto y además para rutinas del robot se pueden tener fórmulas matemáticas que utilicen una buena cantidad de cómputo matemático en punto flotante. Una de las ventajas en este punto es que el robot correrá rutinas y no instrucciones en tiempo real por lo que decrementa el valor de datos que se deben cubrir para realizar cálculos así como

considerablemente se reducen las instrucciones por segundo que debe recibir el robot.

- En tercer lugar se tiene la comunicación con los robots, debido a que el robot Bioloid utiliza comunicación serial y aunque ya se tiene la librería que realiza esta parte se tienen muchos límites en cuanto a cómo usarla por las limitantes que tiene el robot dentro de su micro controlador, para esta parte se busca establecer las instrucciones mínimas que utilicen de lleno esta librería y no tocarlos de ahí en adelante para no tener problemas que puedan comprometer al sistema, en el peor caso se tiene una librería adicional que puede ser sustituida de ser necesario convirtiéndolo en ese momento en un riesgo de alto nivel para el sistema.
- Para el lado de la comunicación con el Kinect se tiene como problemática la cuestión de las librerías ya que no son propiamente automáticas dentro del proyecto, debido a que son del sistema y se deben obtener de manera manual al configurar el proyecto, dentro de las pruebas iniciales se ha visto a veces algunos casos en donde se desconfiguran y se deben modificar las rutas de acceso, volviéndolo un riesgo de nivel medio para el sistema.
- Un riesgo alto es la conversión final del sistema del entorno de desarrollo al entorno de ejecución ya que pueden haber fallas al igual que con el riesgo del Kinect como las rutas de las librerías tanto para el Kinect, como para el Bioloid, además de que una vez montado en el entorno de ejecución, la conexión con el robot no sea la adecuada debido a la selección del puerto debido a que el dispositivo hardware que controla esta parte sea diferente en ese momento y parezca que es error del mismo sistema.

1.6. Descripción General de la Solución Propuesta

El sistema Morpholoid realizará la manipulación de Robots tipos Bioloid mediante el uso de dispositivos de entrada que definan gesturas y puedan ser reconocidas y ejecutadas para varios tipos de robots. Para lograr el objetivo

principal el sistema se verá desarrollado por medio de capas en donde cada una manejará sus propios datos de entrada para mejor segmentación de fallas que pudieran aparecer en el desarrollo. El sistema seguirá el siguiente proceso para funcionar:

1. La primera capa reconocerá una gestura a través del dispositivo de entrada del usuario, en esta capa se generarán todas las ecuaciones que transforman de un sistema de coordenadas o de una entrada de hardware que para el proyecto en específico será el Kinect. El reconocimiento de una gestura dada permitirá poner en acción la cadena que conforma el sistema.
2. Para el segundo paso se tiene la generación de una acción a partir de una gestura dada, permitiendo una gran variedad de movimientos para los robots, esta acción es la acción mayor que imperará sobre el conjunto subsecuente que se generará más adelante.
3. Para el siguiente paso se seleccionará un tipo de robot a partir de la configuración establecida en el sistema y se transformarán las acciones en el reconocimiento de cada una de las partes del robot configurado, una vez que se tenga el mapeo se procederá a ejecutar las acciones en rutinas de instrucciones motor por motor.
4. El penúltimo paso se da en la selección de los motores ya establecidos a las acciones que debe realizar el robot en cada una de sus partes. Dado a que las instrucciones están delimitadas será un cambio sobre algo ya pre configurado y no en tiempo real. En este paso se deberá convertir el conjunto de instrucciones dado en una rutina predefinida que complementará la acción con las instrucciones hacia un robot dado y serán transmitidas para su ejecución.
5. Finalmente se ejecuta un conjunto de instrucciones a bajo nivel dentro del robot que serán visibles físicamente de acuerdo al tipo de robot que se haya configurado para utilizar en el sistema.

Cada conjunto de datos de entrada será aislado dentro de sus mismas capas y hacia las que reciben los mismos tipos de datos para evitar anomalías y

corrupción de datos que puedan llevar a fallas o a ejecuciones de rutinas que no puedan ser interpretadas o ejecutadas por los robots.

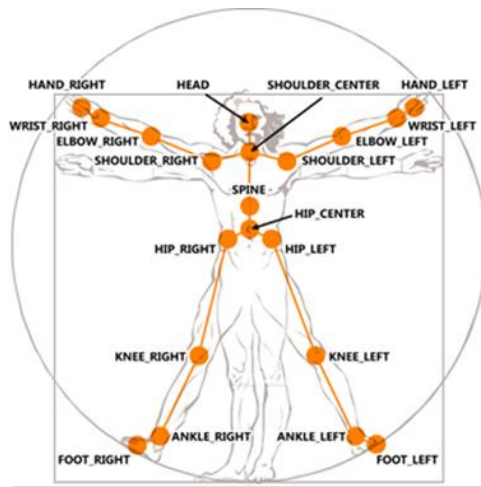
1.7. Tecnologías Utilizadas

1.7.1. C++

Para el desarrollo del proyecto se propone el uso del lenguaje de programación C++ el cual es soportado tanto para el uso del Kinect, como para la comunicación en Serial con el robot Bioloid.

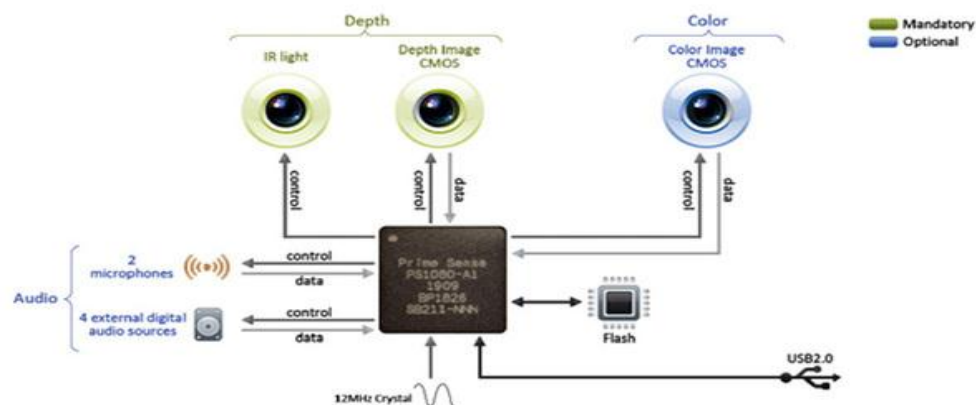
1.7.2. Kinect

El dispositivo de juego para Interfaces Naturales que servirá como la entrada para las pruebas del proyecto y la generación de gesturas de Interfaz Natural con el usuario. El Kinect provee un algoritmo de reconocimiento del cuerpo en forma de esqueleto. Este esqueleto viene delimitado de la siguiente manera:



Cada uno de los puntos del cuerpo viene representado por un eje de coordenadas dado por el mismo Kinect que es interpretado para los ejes: x, y, z. Permitiendo la manipulación una gran variedad de posiciones. El Kinect cuenta con una cámara simple que utiliza para reconocer los elementos reconocibles en primer plano, además cuenta con una segunda cámara que provee de reconocimiento de luz infrarroja y es un proyecto de IR el cual envía al lugar donde se está utilizando cerca de mil punto los

cuales permiten reconocer la profundidad y si son orgánicos los objetos o no, a continuación se presenta un diagrama que muestra la composición de dichas cámaras:



1.7.3. Bioloid

Finalmente se utilizará el robot Bioloid desarrollado por la compañía ROBOTIS, este robot tiene como significado Bio debido a la gran cantidad de transformaciones a robots bípedo-humanoides que se le pueden hacer, este robot se compone de tres componentes



esenciales, en primer lugar está el CM-5 que funciona como el cerebro del robot y es el que recibe las instrucciones. Debido a limitantes en el micro controlador del CM-5 la única manera de establecer comunicación directa desde otros lenguajes de programación, se debe activar el modo de instrucciones dentro del micro controlador una vez hechos los ajustes iniciales de la comunicación serial que utiliza. Una vez realizada la comunicación con el CM-5, este automáticamente registra un arreglo de

servo motores los cuales son identificados por un ID único que va de 0-99 y están hechos en el modelo AX-12. Finalmente el último componente esencial son los sensores del robot que son delimitados de la misma manera que los servo motores, con la diferencia que sus ID van de 100-199 y que su número de modelo es el AX-S1.



1.8. Componentes de Software Integrados

1.8.1. Visual Studio

Como entorno de desarrollo se utilizará Visual Studio en su versión 2012, el cual ya ha sido probado con cada una de las librerías externas de manera individual tanto para el Kinect, como para el Bioloid.

1.8.2. NUI Library

La librería oficial de Microsoft para desarrollo con el Kinect que permite de una manera sencilla utilizarlo en dos lenguajes de programación: C# y C++. La mayor parte de la comunicación y obtención de datos ya está establecida por esta librería y solo basta con iniciar con una comunicación para comenzar a recibir datos de entrada y manipularlos para el sistema que se este desarrollando.

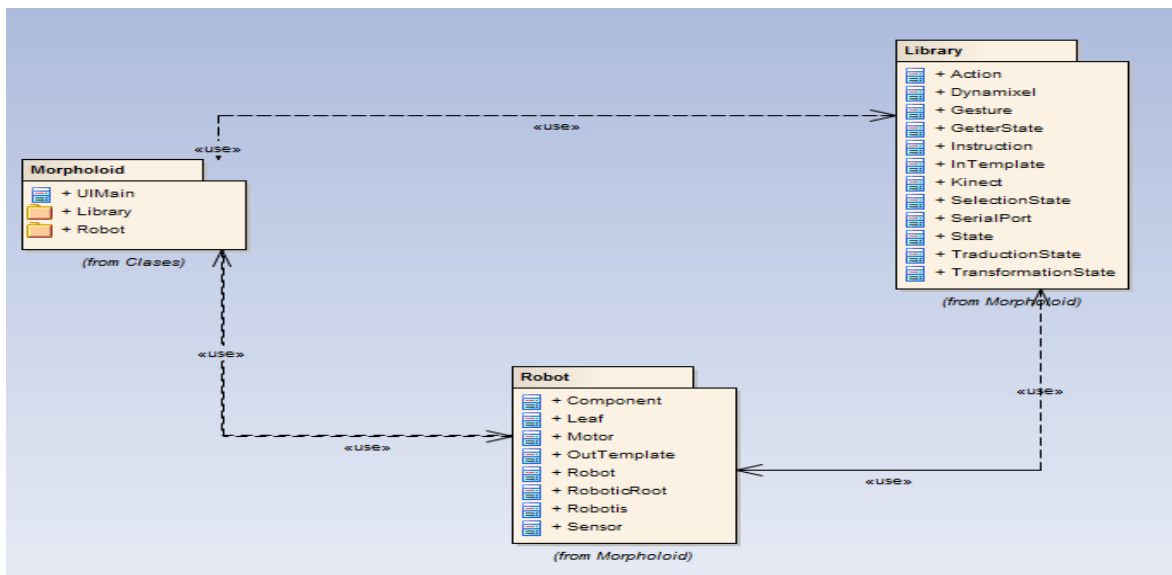
1.8.3. Bioloid Library

Esta librería de comunicación con el Bioloid de ninguna manera es oficial, sin embargo en investigación se ha encontrado que la librería oficial de la compañía ROBOTIS no funciona correctamente debido a limitantes con el micro controlador. Es por esto que esta librería de terceros permite ya

hacer comunicación directa con los servo motores en el desplazamiento y obtención de mensajes de los mismos, se busca poder aumentar el desarrollo de esta librería para poder incorporar mayor flexibilidad así como el manejo de los sensores, sin embargo estos puntos no son esenciales para el proyecto y podrán o no ser desarrollados dependiendo del avance primordial del proyecto.

1.9. Descripción de la Arquitectura General

1.9.1. Arquitectura de Software



El diagrama anterior muestra la forma en que se estructura la comunicación y la arquitectura del sistema Morpholoid, dentro de esta distribución se tiene un paquete global el cual maneja la ejecución principal del sistema así como los sub paquetes que contienen las clases según su agrupación en general la cual se describe a continuación.

Morpholoid:

Este paquete contiene la clase principal que maneja todo el sistema y los sub paquetes Library y Robot que complementan el proceso de la arquitectura del sistema.

Library:

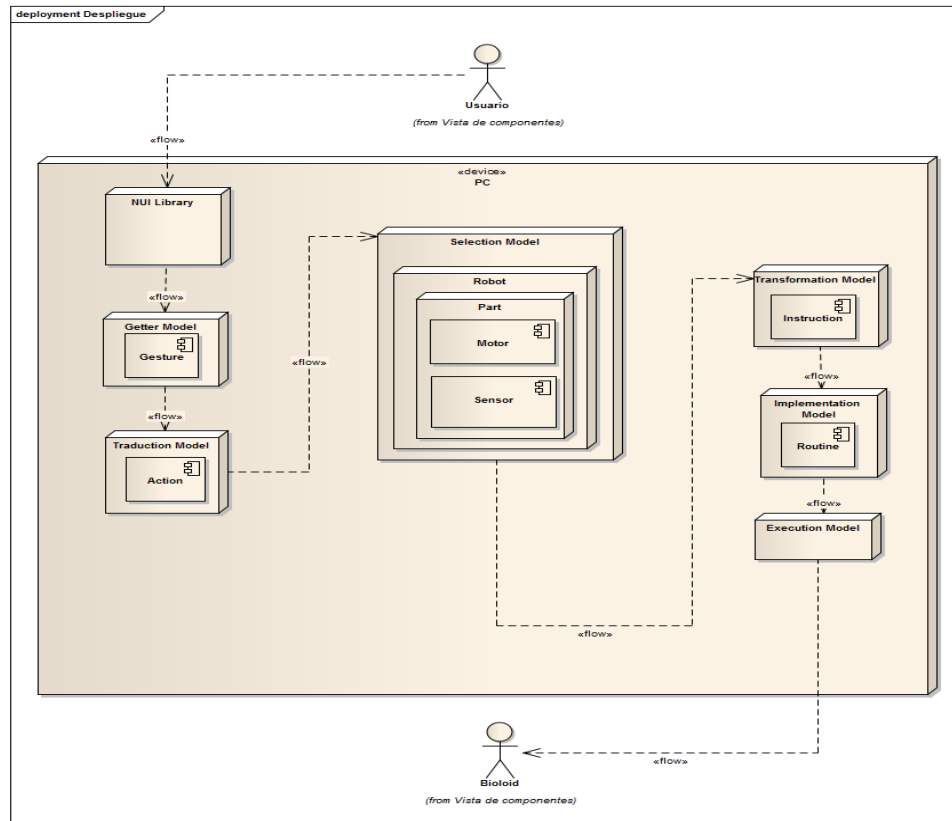
Este sub paquete contiene todas aquellas clases que involucren una llamada hacia una librería externa del proyecto ya sea el compilador, el Kinect o los drivers de comunicación para realizar la conexión serial con los robots, es importante recordar que aunque los robots tienen una estructura la comunicación es totalmente independiente de las instrucciones que estos reciben y puede configurarse fuera del conocimiento de los mismos.

Robot:

Este sub paquete contiene todas aquellas clases que involucren la arquitectura y rutinas de codificación, decodificación de acciones e instrucciones, apoyándose de patrones de diseño para su implementación, comunicación y replicación en información.

Las líneas de comunicación entre cada uno de los paquetes indican su flujo de ida y vuelta ya que para todos los sentidos de alguna manera aunque sea mínima se realiza alguna inter comunicación entre alguna de las clases con otras de paquetes externos a ellas.

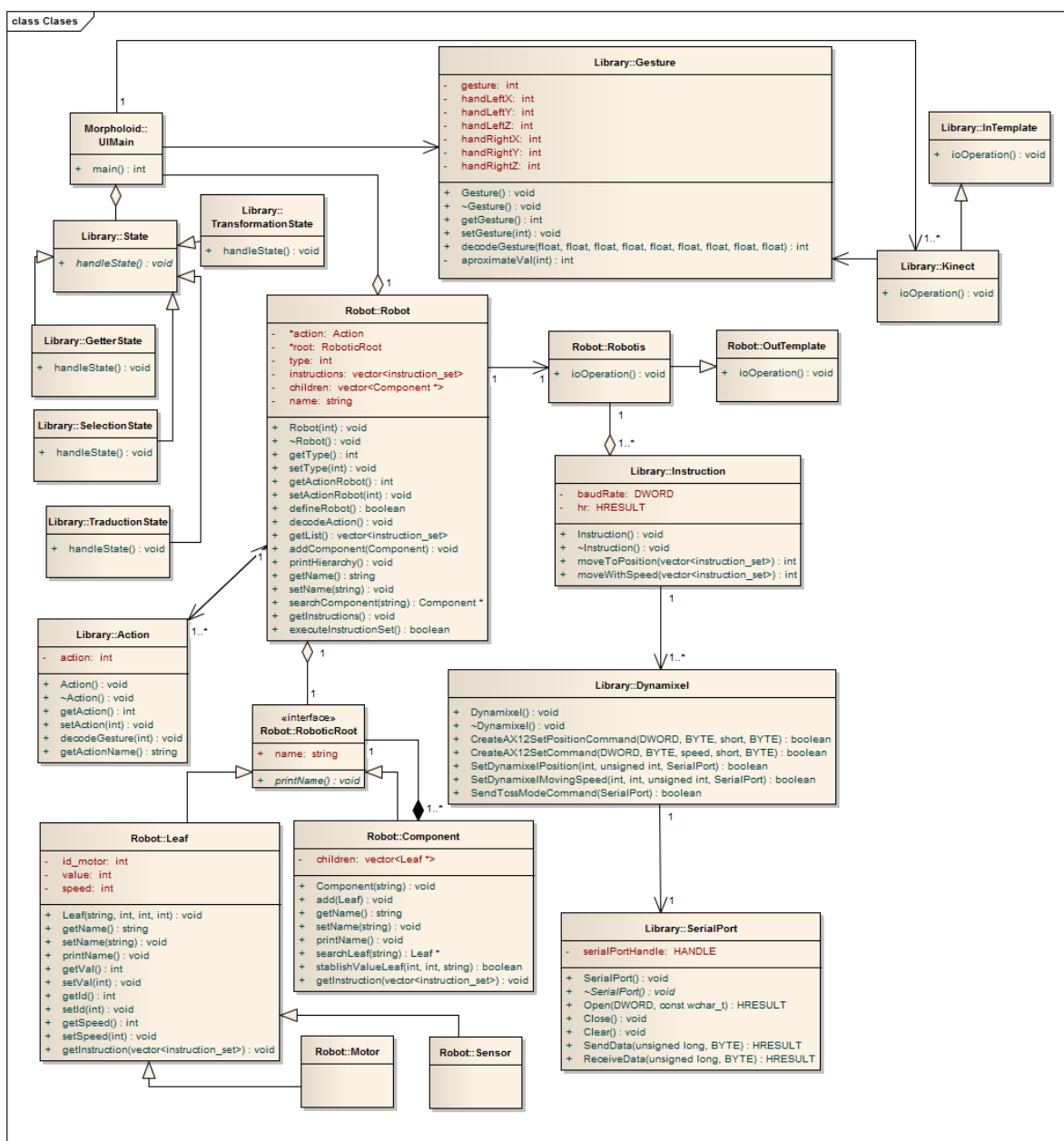
1.9.2. Arquitectura de la Aplicación



El diagrama de despliegue presentado anteriormente muestra dos particularidades importantes del sistema:

1. El proceso en el cual funciona el sistema Morpholoid y la manera en que interactúan cada una de las capas de dicho proceso intercambiando información a sus subsecuentes, este modelo linear permite mostrar el flujo básico de información.
2. El aislamiento por capas permite mostrar la forma mediante la cual el procesos utilizan las capas para distribuir la información, es importante aclarar en este punto que aunque se sigue la estructura de capas marcadas desde la introducción de este documento, la comunicación entre clases al momento de utilizar los patrones de diseño permite de manera más sencilla efectuar este proceso y delimitar de mejor manera las capas aunque sean dependientes sobre algunas clases maestras que manejan la información en general.

1.9.3. Diseño del Software



El diagrama de clases de la página anterior muestra en su totalidad el sistema Morpholoid con sus respectivas conexiones y diseño en cuanto a patrones mostrando por qué existe una comunicación entre todos los paquetes de la aplicación, en total se tienen 10 clases que componen todo el sistema.

A continuación se describe brevemente el propósito de cada una de las clases:

UIMain: Esta clase es el main principal del sistema por el cual corre todo el proceso de Morpholoid, dentro de esta clase se segmentan cada uno de los objetos principales y los datos que se reciben, así como la configuración sobre el robot inicial dentro de la aplicación y también la interacción directa con el Kinect y el reconocimiento del esqueleto en puntos cartesianos.

Gesture: Esta clase se encarga de recibir, codificar y generar a partir de un conjunto de puntos cartesianos una serie de gesturas que están delimitadas por la aplicación.

Action: Esta clase se encarga de recibir, codificar y generar una a partir de una gestura dada una acción equivalente según sea el tipo de robot configurado para el sistema.

Instruction: Esta clase se encarga de convertir un listado general de instrucciones a la forma de bajo nivel que es recibida por el robot.

Dynamixel: Esta clase se encarga de establecer la descomposición de instrucciones de bajo nivel en bits para su envío mediante comunicación serial, esta clase es el puente entre las clases de Instrucciones y Puerto Serial.

SerialPort: Esta clase se encarga de enviar vía comunicación Serial a través de los drivers de Windows las instrucciones en el formato de bajo nivel para el robot.

Robot: Esta clase se encarga de administrar los tipos de robots que contiene configurado el sistema y las diferentes formas de generación de acciones propias de cada robot así como la comunicación para la generación de instrucciones igualmente dependiendo del tipo de robot dado.

RoboticRoot: Esta clase se encarga de ser el primer paso en el patrón de diseño utilizado en el sistema, el cual contiene siendo una interfaz los métodos propios que tendrán que tener las clases subsecuentes.

Component: Esta clase se encarga de generar partes propias de un robot que contienen un conjunto de hojas que pueden estar establecidos como motores y/o

sensores, aquí se permite buscar un id específico para guardar una instrucción a alto nivel.

Leaf: Esta clase se encarga de representar la forma de más bajo nivel de un robot usando un alto nivel para establecer los valores que después serán traducidos en instrucciones.

1.10. Patrones del Diseño

1.10.1. Descripción del Patrón

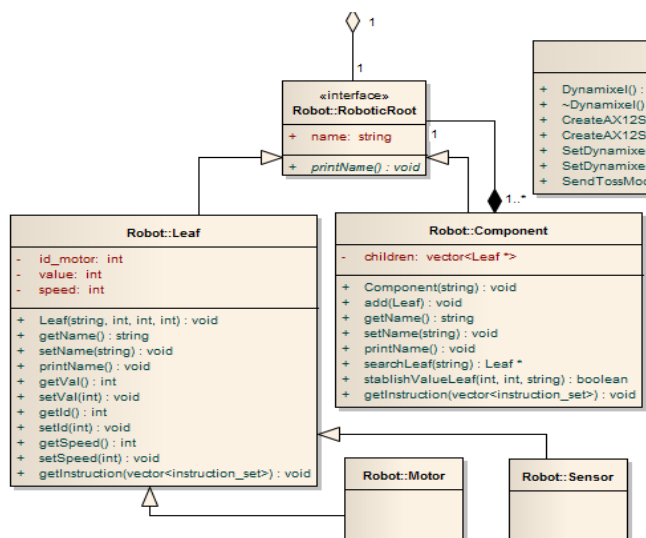
El patrón de diseño que se utiliza para el sistema es el Composite, este patrón es de tipo estructural y permite agrupar objetos en estructuras más complejas haciendo que la búsqueda y manipulación de los objetos más simples ayude a dar poder a estructuras más complejas y grandes.

1.10.2. Aplicación

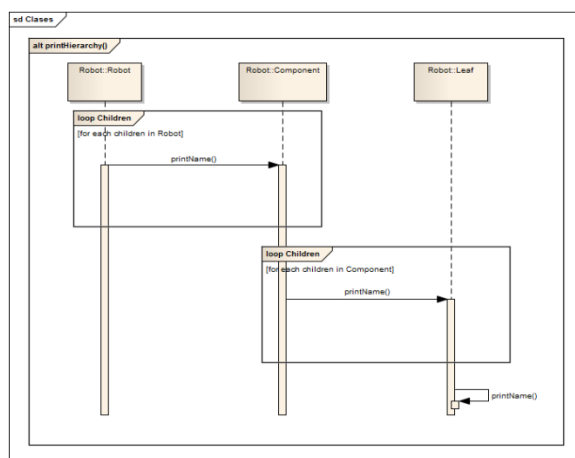
Este patrón se eligió en uso para el sistema ya que la configuración física de los robots Bioloid se compone de un conjunto de motores y sensores que ensamblados crean una red local entre ellos. Siguiendo los planos establecidos en los manuales de tipos de robots del Kit de robótica del Bioloid cada uno de los motores utilizados debe contener un id específico permitiendo que no solo la red local de motores, sino que la identificación de cada uno sea más sencilla. El uso del composite para simular este comportamiento de armado de los robots permite establecer segmentos entre las partes de un robot siendo más sencillo de ver el caso del robot humanoide, Utilizando el composite se pueden establecer componentes o en este caso, partes del cuerpo del robot, a su vez cada parte contiene un conjunto de motores y sensores. Utilizando los métodos que se pasan a través de la interfaz del composite se pueden generar estructuras de control en simulación bastante complejas para realizar algoritmos desde nivel simple hasta muy complicado, tal es el caso de la realización de la caminata del

robot, la cual al segmentar las piernas permite delimitar de manera independiente como se mueve cada parte pero llegando al nivel de detalle de mover un solo motor ligeramente que puede influir en el equilibrio total del robot.

1.10.3. Diagrama de Clases del Patrón



1.10.4. Diagrama de Secuencia del Patrón



1.10.5. Descripción del Patrón

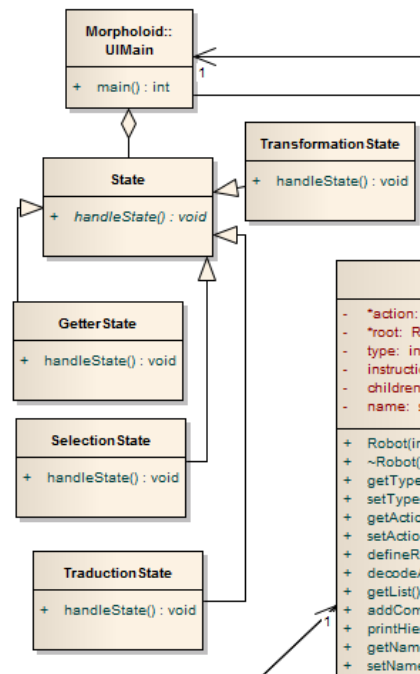
Otro patrón de diseño que se utiliza en el sistema es el State, este patrón permite manejar el comportamiento del sistema dependiendo

del estado del mismo. En el caso del sistema se tiene un sistema de capas, cuando cada capa avanza se requiere hacer una acción específica para cada una de ellas, y de esto se encarga el State.

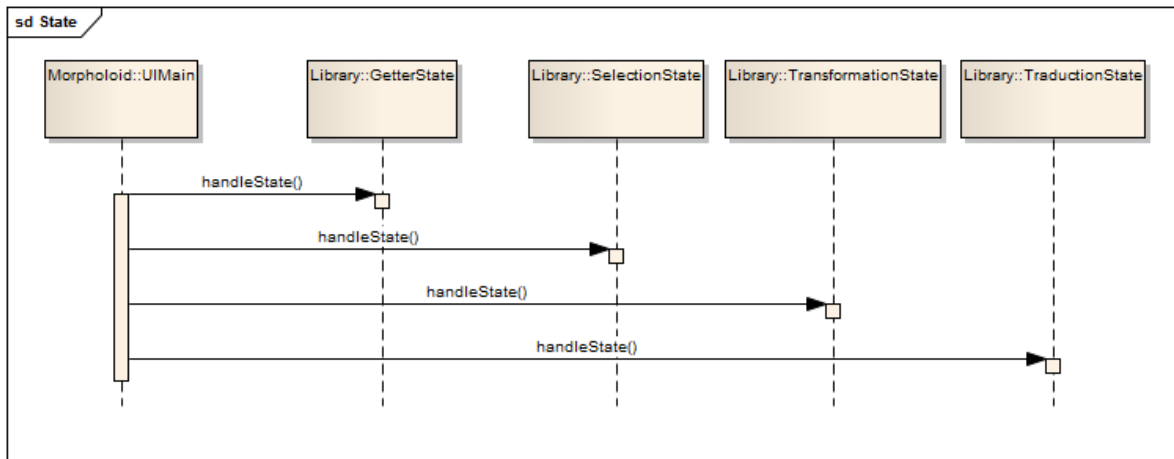
1.10.6. *Aplicación*

Este patrón se eligió debido a que permite incorporar la estructura de capas de manera más sencilla y mantener los atributos de calidad de extensibilidad disponibles y en funcionamiento, así si se requiere agregar una nueva capa solo basta con agregar un nuevo estado que maneje los nuevos datos.

1.10.7. *Diagrama de Clases del Patrón*



1.10.8. Diagrama de Secuencia del Patrón



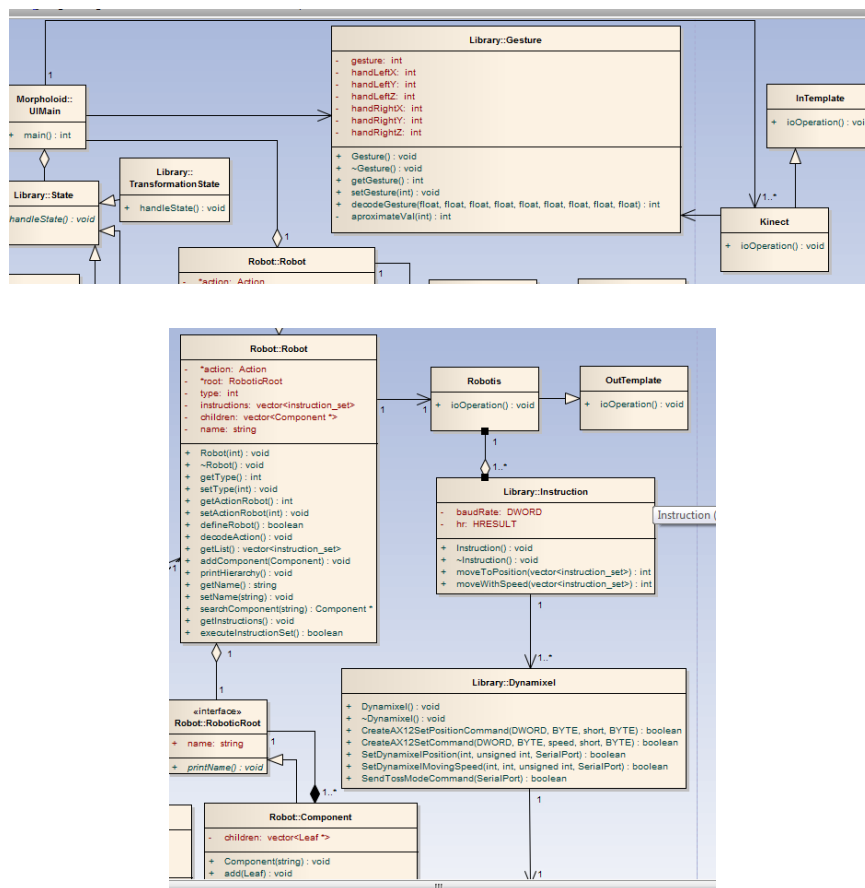
1.10.9. Descripción del Patrón

Finalmente se usó un mismo patrón de diseño 2 veces para segmentar la entrada y salida del sistema. El patrón usado fue el Template, este patrón permite generar una plantilla con herencia sobre un grupo en común. Este patrón se encuentra dentro de los patrones de comportamiento y opera mediante una superclase y a su vez subclases que heredan un algoritmo o comportamiento en específico.

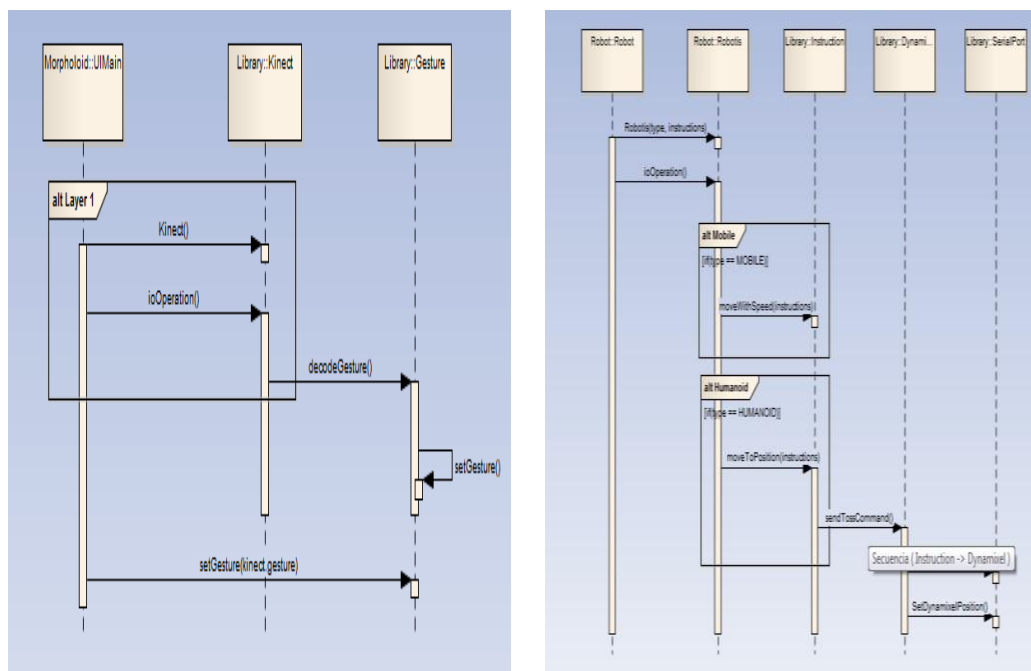
1.10.10. Aplicación

Este patrón se utilizó para modelar la entrada y salida de datos, esto para permitir extensibilidad en dispositivos de entrada, actualmente se tiene configurado el kinect, sin embargo se pueden generar más modelos que permiten el mismo comportamiento con el cual se utiliza el Kinect. De la misma manera la salida de datos permite manejar robots de diferente tipo tan solo con utilizar el tipo de comunicación específica con las instrucciones según el tipo de robot que se este utilizando.

1.10.11. Diagrama de Clases del Patrón



1.10.12. Diagrama de Secuencia del Patrón



2. Evaluación de la Arquitectura de Software

2.1. Técnica de Evaluación Utilizada

Para la evaluación de la arquitectura se decidió utilizar el Método de Evaluación para Arquitecturas Basadas en Componentes (MECABIC), el cual está inspirado en el Architecture Tradeoffs Analysis Method (ATAM), ABD, ARID, Losavio, Bosch, propuesto por Carrascoso, Chaviano y Céspedes. MECABIC se enfoca en evaluar y analizar la calidad exigida por los usuarios del sistema basado en componentes.

Procedimiento de Evaluación

El método MECABIC propone tres equipos que estarán involucrados en las diferentes fases de la evaluación.

Equipo	Definición	Fases en las que participan
Arquitectos	Responsables de generar y documentar una Arquitectura de Software para el sistema estudiado.	
Evaluador	Integrado por personas expertas en asuntos de calidad quienes guiarán el proceso de evaluación de la arquitectura.	Todas
Relacionados	Son las personas involucradas de alguna manera con el sistema: programadores, usuarios, gerentes, entre otros.	Fases 1, 3 y 4.

Para el caso del sistema se tiene la siguientes distribución:

Arquitecto: Diseñador del sistema.

Evaluador: Profesor titular de la materia.

Relacionados: Personas que han utilizado el Kinect y los Bioloid y que servirían de usuarios finales del sistema.

Las fases del método son las siguientes:

1. Presentación: consiste en dar a conocer el método a todos los grupos involucrados. Se presenta en que consiste el método a todos los involucrados. Se espera que todos comprendan el método y el objetivo de este. También se presenta la arquitectura a evaluar y las características de calidad esperadas.

2. Investigación y Análisis: en esta fase se determina como se va a estudiar la arquitectura.

a. Generación del árbol de utilidad: está compuesto por un conjunto de escenarios de interés que se esperan obtener de los aspectos de calidad de la arquitectura.

b. Selección de escenarios: el grupo evaluador presenta los diferentes escenarios a considerar al resto de los participantes y en conjunto se eligen cuales serán incluidos en el árbol de utilidad.

c. Priorización: para los escenarios no contemplados se deberán de priorizar para así poder determinar cuales se incluirán en el árbol de utilidad.

3. Prueba: consiste en la revisión de la segunda fase.

1. Revisión del árbol de utilidad: se evalúa si los escenarios planteados fueron correctos y si existe la necesidad de modificarlos.

2. Revisión de los elementos de diseño definidos: se evalúa si los escenarios promueven los atributos de calidad propuestos, de no ser así, se deberá reconsiderarlos.

4. Presentación de Resultados: se dan a conocer a todos los interesados los resultados de la evaluación.

Árbol de Utilidad

Atributo de Calidad	Característica	Escenario	Prioridad
Funcionalidad	Interoperabilidad	Se establece un sistema de capas, se busca que el sistema siga este proceso estrictamente en su ejecución principal.	Alta
Confiabilidad	Tolerancia a fallas	En caso de tener una configuración errónea de robot al que se tiene conectado físicamente el sistema no deberá terminarse abruptamente.	Media
	Tolerancia a fallas	El sistema deberá notificar al usuario cuando no se pueda efectuar correctamente una operación a alguno de los motores del robot.	Baja
	Tolerancia a fallas	El sistema deberá limitar los movimientos de motores según ciertos tipos de robots para no dañar a los mismos.	Media
	Tolerancia a fallas	El sistema deberá estar preparado para cuando se no haya conectividad debido a la comunicación serial.	Alta
Eficiencia	Utilización de recursos	Al realizar la lectura del Kinect debe mantenerse	Alta

		una fluidez en la ejecución del sistema.	
Mantenibilidad	Acoplamiento	Cada capa del sistema deberá recibir y regresar los datos propios que cada una propone para efectos de mantener la arquitectura propuesta.	Alta
Portabilidad	Modularidad	Se permite la extensión de nuevos módulos al sistema o nuevas capas para la extensión en base a la arquitectura propuesta.	Alta
	Modularidad	Se permite la extensión de nuevos sub módulos pertenecientes a cada capa para la extensión de la funcionalidad del sistema.	Media
	Reemplazabilidad	Se permite modificar los módulos ya existentes para mejorar las rutinas dadas para ciertos robots.	Media

Diseño y Ejecución de Pruebas

Prueba 1.1 Se hace un flujo total del proceso de capas en orden normal.

Proceso

- 1.-Iniciar el programa
- 2.-Seleccionar un tipo de robot
- 3.-El usuario hace gestura de introducción de instrucción natural
- 4.-El usuario hace gestura de instrucción para acción a Robot

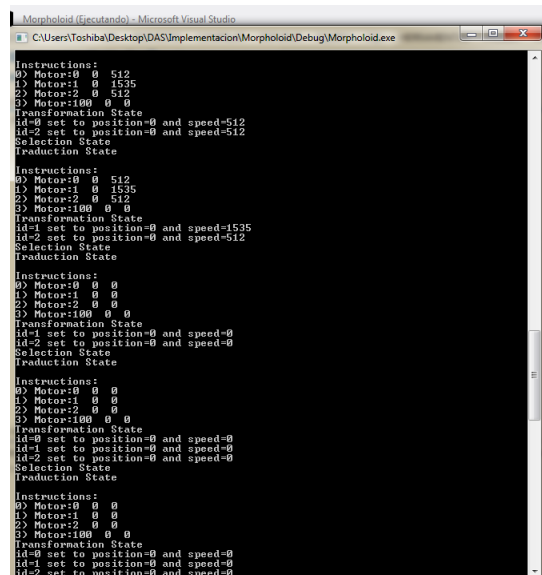
Responsable de ejecución: Arquitecto

Resultados Esperados

Se ejecuta la instrucción dada correctamente en el robot configurado.

Resultados Obtenidos

Se ejecutó el procesos obteniendo como resultado el esperado sobre ejecutar correctamente la instrucción en el robot configurado, sin embargo el análisis de código dejo ver que aún se debe implementar el patrón de diseño para que la extensibilidad del sistema sea el adecuado.



```
Instructions:
0) Motor:0 0 512
1) Motor:1 0 1535
2) Motor:2 0 512
3) Motor:100 0 0
Transformation State
id=0 set to position=0 and speed=512
id=2 set to position=0 and speed=512
Selection State
Production State
Instructions:
0) Motor:0 0 512
1) Motor:1 0 1535
2) Motor:2 0 512
3) Motor:100 0 0
Transformation State
id=1 set to position=0 and speed=1535
id=2 set to position=0 and speed=512
Selection State
Production State
Instructions:
0) Motor:0 0 0
1) Motor:1 0 0
2) Motor:2 0 0
3) Motor:100 0 0
Transformation State
id=1 set to position=0 and speed=0
id=2 set to position=0 and speed=0
Selection State
Production State
Instructions:
0) Motor:0 0 0
1) Motor:1 0 0
2) Motor:2 0 0
3) Motor:100 0 0
Transformation State
id=0 set to position=0 and speed=0
id=1 set to position=0 and speed=0
id=2 set to position=0 and speed=0
Selection State
Production State
Instructions:
0) Motor:0 0 0
1) Motor:1 0 0
2) Motor:2 0 0
3) Motor:100 0 0
Transformation State
id=0 set to position=0 and speed=0
id=1 set to position=0 and speed=0
id=2 set to position=0 and speed=0
Selection State
Production State
```


Prueba 1.2 Se hace un flujo normal haciendo cambios en el gestor de entrada (Kinect)

Proceso

- 1.-Desconectar el Kinect de la computadora
- 2.-Iniciar el programa

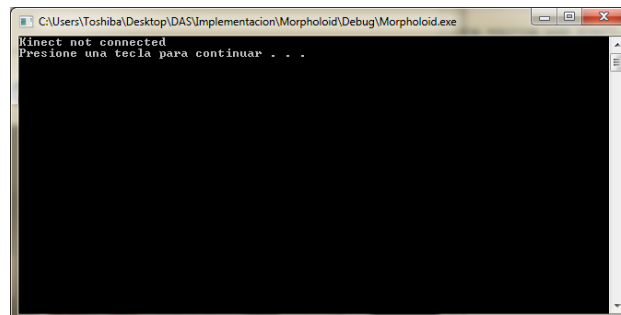
Responsable de ejecución: Arquitecto

Resultados Esperados

El sistema deberá notificar al usuario que no hay un dispositivo Kinect conectado y deberá terminar correctamente.

Resultados Obtenidos

La prueba se ejecutó correctamente sin embargo para la extensibilidad del sistema se dará un poco más de flexibilidad sobre cuáles son los dispositivos de entrada.



Prueba 1.3 Se hace un flujo normal haciendo cambios en el gestor de salida (Robot)

Proceso

- 1.-Desconectar el robot de la computadora
- 2.-Iniciar el programa
- 3.-Seleccionar un tipo de robot
- 4.-El usuario hace gestura de introducción de instrucción natural
- 5.-El usuario hace gestura de instrucción para acción a Robot

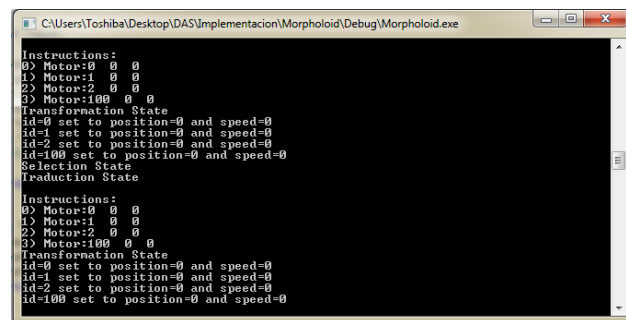
Responsable de ejecución: Arquitecto

Resultados Esperados

El sistema se ejecutará correctamente pero al no haber robot conectado no se verá un medio de salida. El sistema no deberá fallar al realizar la desconexión.

Resultados Obtenidos

La prueba se ejecutó correctamente, los resultados físicos de la prueba demostraron que al no haber un robot conectado el programa sigue ejecutándose ya que no hay manera de saber que no hay un dispositivo de salida conectado. Sin embargo a pesar de este estado se encontró que el sistema no sufre de anomalías y que al contrario el sistema sigue creyendo que el robot está conectado.



```
C:\Users\Toshiba\Desktop\DAS\Implementacion\Morpholoid\Debug\Morpholoid.exe

Instructions:
0) Motor:0 0 0
1) Motor:1 0 0
2) Motor:2 0 0
3) Motor:100 0 0
Transformation State
id=0 set to position=0 and speed=0
id=1 set to position=0 and speed=0
id=2 set to position=0 and speed=0
id=100 set to position=0 and speed=0
Selection State
Transformation State
Instructions:
0) Motor:0 0 0
1) Motor:1 0 0
2) Motor:2 0 0
3) Motor:100 0 0
Transformation State
id=0 set to position=0 and speed=0
id=1 set to position=0 and speed=0
id=2 set to position=0 and speed=0
id=100 set to position=0 and speed=0
```

Prueba 2.1 Se configura un robot diferente para un flujo normal del sistema

Proceso

- 1.-Iniciar el programa
- 2.-Seleccionar un tipo de robot
- 3.-El usuario hace gestura de introducción de instrucción natural
- 4.-Desconectar el robot de la computadora
- 5.-El usuario hace gestura de instrucción para acción a Robot diferente al configurado.

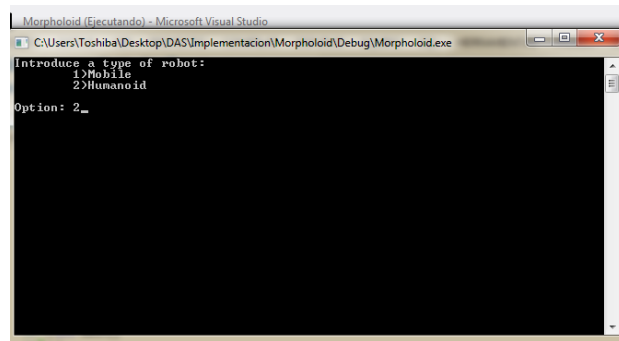
Responsable de ejecución: Relacionados

Resultados Esperados

Debido a que la acción de interacción física con el robot corre de manera paralela al sistema, el resultado de la ejecución al robot debe ser correcto aunque resulte en un movimiento diferente al esperado ya que no es el correcto, para el caso del sistema se espera que se ejecute correctamente.

Resultados Obtenidos

La prueba se ejecutó correctamente haciendo dos variantes sobre los robots previamente configurados, móvil y humanoide. En el primer escenario se configuró para el sistema un robot móvil, mientras que se conectó físicamente el robot humanoide. En esta prueba debido a que el robot es diferente y la configuración de los motores es diferente el sistema no puede introducir las instrucciones al robot. Para el caso inverso, se configuró el robot humanoide con el robot móvil físicamente conectado, esto permitió que el robot se moviera de manera diferente a la configurada ya que los motores aunque soportan en el modo la configuración no son los reales que se deben usar. Para el caso del sistema no se detectaron anomalías que hicieran que fallará o que se corrompiera durante su ejecución.



Prueba 2.2 Se desconecta el robot durante un flujo normal del sistema

Proceso

- 1.-Iniciar el programa
- 2.-Seleccionar un tipo de robot
- 3.-El usuario hace gestura de introducción de instrucción natural
- 4.-Desconectar el robot de la computadora
- 5.-El usuario hace gestura de instrucción para acción a Robot

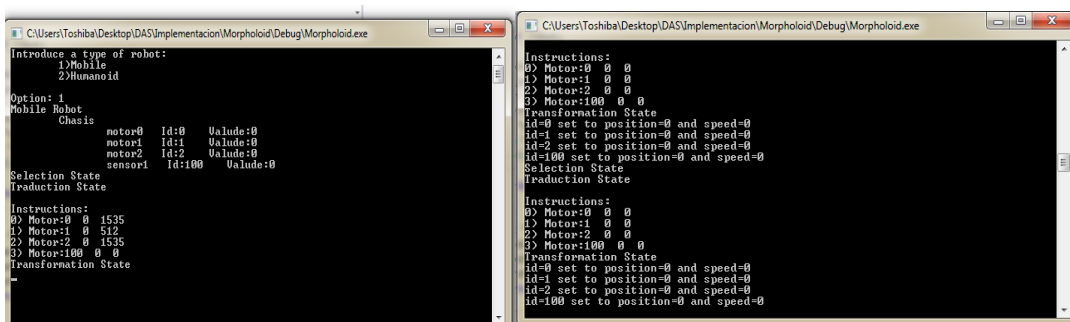
Responsable de ejecución: Relacionados

Resultados Esperados

El sistema deberá soportar la desconexión del robot sin hacer cambios aparentes en el flujo del mismo.

Resultados Obtenidos

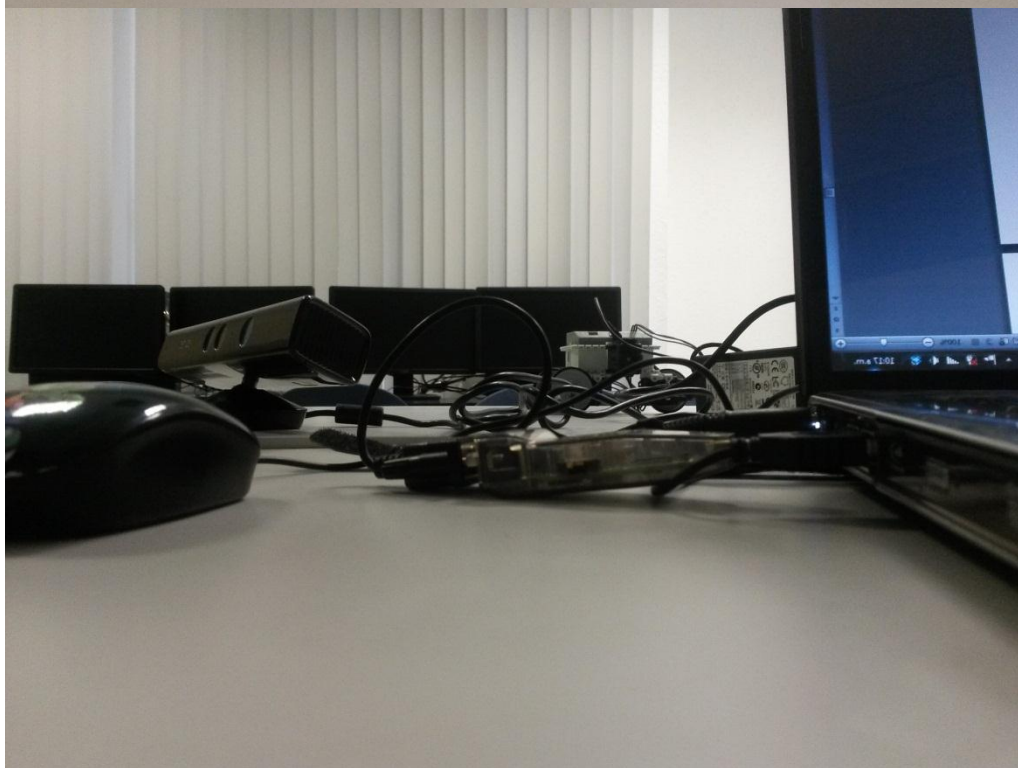
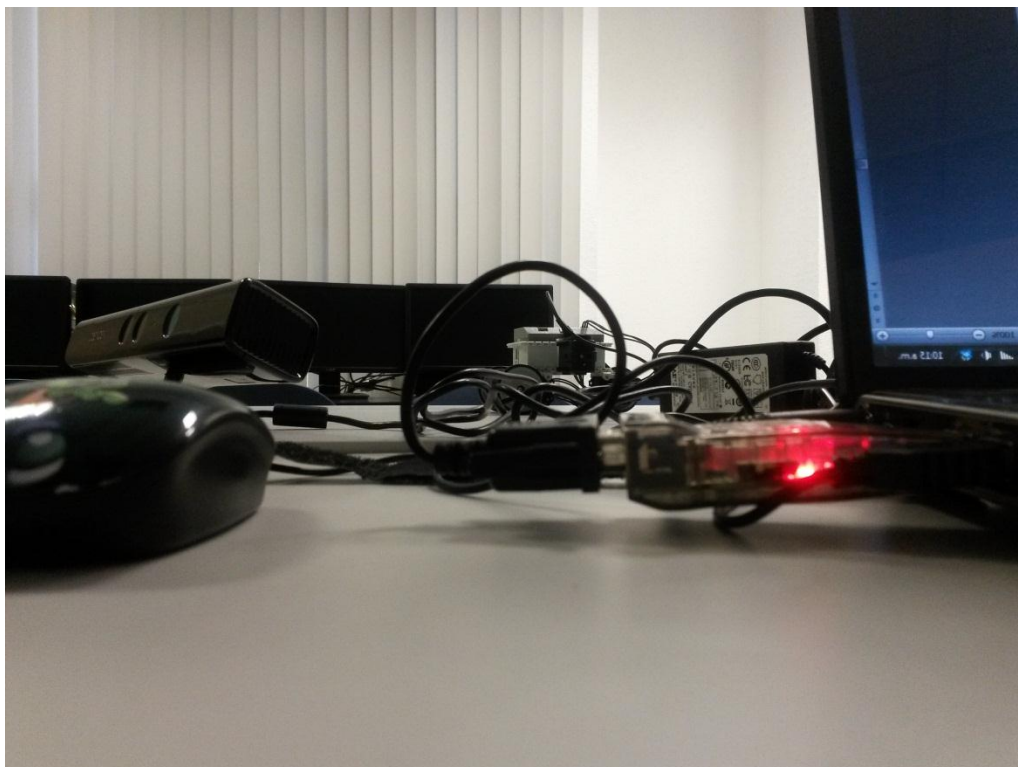
La prueba se ejecutó correctamente, el robot al ser desconectado no presentó problemas al sistema, sin embargo en el caso del robot móvil si se deja en ejecución debe volver a conectarse, debido a que el modo de rueda queda activo a nivel del robot.



```
C:\Users\Toshiba\Desktop\IAS\Implementacion\Morpholoid\Debug\Morpholoid.exe
Introduce a type of robot:
1) Mobile
2) Humanoid

Option: 1
Mobile Robot
Chassis
  motor0 Id:0 Value:0
  motor1 Id:1 Value:0
  motor2 Id:2 Value:0
  sensor1 Id:100 Value:0
Selection State
Introduction State
Instructions:
0) Motor:0 0 1535
1) Motor:1 0 512
2) Motor:2 0 1535
3) Motor:100 0 0
Transformation State

C:\Users\Toshiba\Desktop\IAS\Implementacion\Morpholoid\Debug\Morpholoid.exe
Instructions:
0) Motor:0 0 0
1) Motor:1 0 0
2) Motor:2 0 0
3) Motor:100 0 0
Transformation State
id=0 set to position=0 and speed=0
id=1 set to position=0 and speed=0
id=2 set to position=0 and speed=0
id=100 set to position=0 and speed=0
Selection State
Introduction State
Instructions:
0) Motor:0 0 0
1) Motor:1 0 0
2) Motor:2 0 0
3) Motor:100 0 0
Transformation State
id=0 set to position=0 and speed=0
id=1 set to position=0 and speed=0
id=2 set to position=0 and speed=0
id=100 set to position=0 and speed=0
```



Prueba 2.3 Se envía una instrucción que no pueda ser reconocida por el robot configurado

Proceso

- 1.-Configurar una instrucción que no pueda ser realizada por el robot
- 2.-Ejecutar el programa
- 3.-Seleccionar un tipo de robot
- 4.-El usuario hace gestura de introducción de instrucción natural
- 5.-Desconectar el robot de la computadora
- 6.-El usuario hace gestura de instrucción para acción a Robot

Responsable de ejecución: Arquitecto

Resultados Esperados

El sistema deberá soportar la instrucción mal configurada y dependiendo del robot o no ejecutará la carga de la instrucción o se ejecutará un movimiento diferente en el robot al normal, sin comprometer al sistema.

Resultados Obtenidos

La prueba se ejecutó correctamente se presentan las alteraciones hechas al robot:

```

140 //
141 switch(getType()){
142     case MOBILE:
143         Component *chasis;
144         switch (action->getAction())
145         {
146             case CENTER_SELF:
147                 chasis = searchComponent("Chasis");
148                 if(chasis != NULL){
149                     chasis->establishValueLeaf("motor0", 0, 0);
150                     chasis->establishValueLeaf("motor1", 0, 0);
151                     chasis->establishValueLeaf("motor2", 0, 0);
152                     //printlnHierarchy();
153                 }
154                 break;
155             case MOVEMENT_UP:
156                 chasis = searchComponent("Chasis");
157                 if(chasis != NULL){
158                     chasis->establishValueLeaf("motor0", 0, 512);
159                     chasis->establishValueLeaf("motor1", 0, 1535);
160                     chasis->establishValueLeaf("motor2", 0, 512);
161                     //printlnHierarchy();
162                 }
163                 break;
164             case MOVEMENT_DOWN:
165                 chasis = searchComponent("Chasis");
166                 if(chasis != NULL){
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
```

Prueba 2.4 Enviar valores inválidos para motores que no estén en modo llanta

Proceso

- 1.-Configurar el robot móvil en el sistema y conectar físicamente el robot móvil.
- 2.-El usuario hace gestura de introducción de instrucción natural
- 3.-Desconectar el robot de la computadora
- 4.-El usuario hace gestura de instrucción para acción a Robot

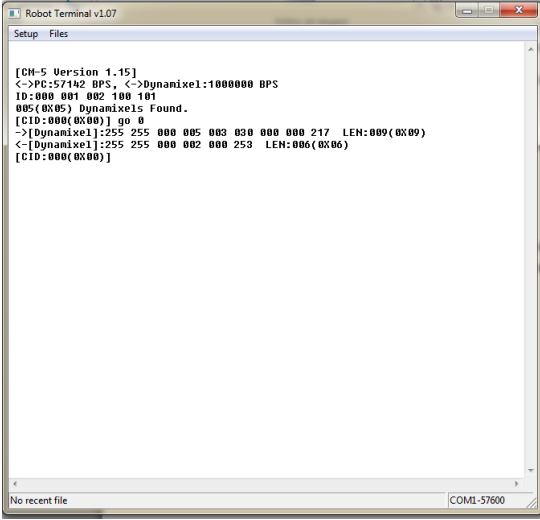
Responsable de ejecución: Arquitecto

Resultados Esperados

El sistema no deberá introducir la instrucción al robot ya que los motores que no están en modo rueda no pueden tomar el tipo de instrucciones.

Resultados Obtenidos

Se ejecutó satisfactoriamente la prueba.



```
Robot Terminal v1.07
Setup  Files

[CH-5 Version 1.15]
<->PC:57142 BPS, <->Dynamixel:1000000 BPS
ID:000 001 002 100 101
005(0X05) Dynamixel Found.
[CMD:000(0X00)] go 0
->[Dynamixel]:255 255 000 005 003 030 000 000 217 LEN:009(0X09)
<-[Dynamixel]:255 255 000 002 000 253 LEN:006(0X06)
[CMD:000(0X00)]
```


Prueba 2.5 Se envía una instrucción cuando no haya robot conectado

Proceso

- 1.-Configurar el robot móvil en el sistema y desconectar físicamente cualquier robot.
- 2.-El usuario hace gestura de introducción de instrucción natural
- 3.-Desconectar el robot de la computadora
- 4.-El usuario hace gestura de instrucción para acción a Robot

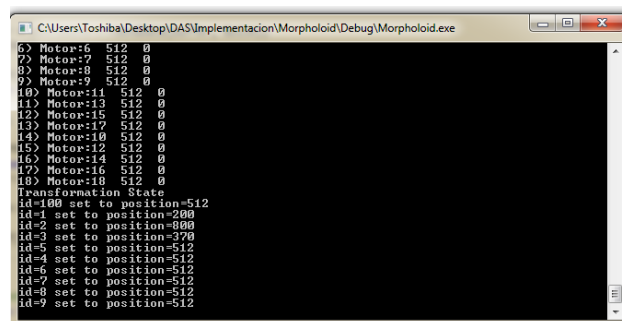
Responsable de ejecución: Arquitecto

Resultados Esperados

El sistema deberá intentar escribir la instrucción sin éxito y notificar al usuario.

Resultados Obtenidos

La prueba se ejecutó correctamente, el usuario es retroalimentado por el sistema que no se puede establecer una comunicación serial con un dispositivo de salida.



```
C:\Users\Toshiba\Desktop\DAS\Implementacion\Morpholoid\Debug\Morpholoid.exe
6> Motor:6 512 0
7> Motor:7 512 0
8> Motor:8 512 0
9> Motor:9 512 0
10> Motor:11 512 0
11> Motor:13 512 0
12> Motor:15 512 0
13> Motor:17 512 0
14> Motor:18 512 0
15> Motor:12 512 0
16> Motor:14 512 0
17> Motor:16 512 0
18> Motor:18 512 0
Transformation State
id=180 set to position=512
id=1 set to position=2000
id=2 set to position=8000
id=3 set to position=3700
id=5 set to position=512
id=4 set to position=512
id=6 set to position=512
id=7 set to position=512
id=8 set to position=512
id=9 set to position=512
```

Prueba 3.1 Se ejecuta el proceso de capas sin lags de delay para realizar instrucciones en el robot

Proceso

- 1.-Configurar el robot móvil en el sistema y desconectar físicamente cualquier robot.
- 2.-El usuario hace gestura de introducción de instrucción natural
- 3.-Desconectar el robot de la computadora
- 4.-El usuario hace gestura de instrucción para acción a Robot

Responsable de ejecución: Relacionados

Resultados Esperados

El sistema deberá ejecutar un ciclo de instrucciones continuo sin detenerse convirtiendo la gestura en instrucciones del robot de forma natural que el usuario pueda ver.

Resultados Obtenidos

Se ejecutó de manera correcta la prueba obteniendo resultados muy buenos sobre la comunicación en tiempo real, esta prueba en especial permitió ver el alcance hacia nuevas áreas de investigación en el sistema que permitan algoritmos en tiempo real que sean más complejos usando la misma arquitectura ya hecha.

*Se anexa video

Prueba 3.1 Se ejecuta el proceso de capas con tiempo suficiente para ejecutar instrucciones por separado que sean distinguibles por el usuario

Proceso

- 1.-Configurar el robot móvil en el sistema y desconectar físicamente cualquier robot.
- 2.-El usuario hace gestura de introducción de instrucción natural
- 3.-Desconectar el robot de la computadora
- 4.-El usuario hace gestura de instrucción para acción a Robot

Responsable de ejecución: Relacionados

Resultados Esperados

El sistema deberá mostrar aparentemente al usuario que ejecuta instrucciones pausadas sobre el mismo tipo de gestura dejando ver que se ejecuta correctamente.

Resultados Obtenidos

La prueba se ejecutó correctamente ante varios tipos de velocidad en usuarios, en primer lugar un usuario lento el cual pausaba los movimientos para ver como cambiaba la instrucción, en segundo lugar un usuario de velocidad media que movía al robot de manera natural observando los cambios pero con algunos cambios repentinos sobre la instrucción dada; finalmente un usuario veloz que apenas dejara ver la instrucción dada. Los resultados fueron: para el usuario lento se ejecutaron correctamente todas y cada una de las instrucciones, para el usuario medio de igual manera se ejecutaron correctamente todas las instrucciones dadas, para el usuario veloz se tiene que aunque se ejecuten correctamente las instrucciones a veces se tiene una de sincronía del robot tomando la última reconocida y sin fallas en el sistema.

*Se anexa video.

Prueba 4.1 Se evalúa la capa getter para las gesturas

Proceso

- 1.-Segmentar el sistema a la capa getter.
- 2.-Evaluar el comportamiento y observar si es el comportamiento deseado.

Responsable de ejecución: Arquitecto

Resultados Esperados

Se espera que dependiendo del dispositivo de entrada configurado Kinect, se obtenga una gestura.

Resultados Obtenidos

Para la capa getter se obtuvo un resultado satisfactorio al aislar por completo la capa y observar su comportamiento a detalle, se tienen todas las gesturas programadas así como extensibilidad sobre las mismas.

```
72 if(layer == LAYER_1){
73     kinect->ioOperation();
74     gestures->setGesture(kinect->gesture->getGesture());
75
76     switch(gestures->getGesture()){
77         case HANDS_UP:
78             layer = LAYER_2;
79             break;
80         case HANDS_DOWN:
81             layer = LAYER_2;
82             break;
83         case HORIZONTAL_HANDS:
84             layer = LAYER_2;
85             break;
86         case VERTICAL_HANDS:
87             layer = LAYER_2;
88             break;
89         case DOWN_L_RIGHT_HANDS:
90             layer = LAYER_2;
91             break;
92         case DOWN_L_LEFT_HANDS:
93             layer = LAYER_2;
94             break;
95         default:
96             //cout << "Gesture not defined" << endl;
97             break;
98     }
```

Prueba 4.2 Se evalúa la capa selection para la acción

Proceso

- 1.-Segmentar el sistema a la capa selection.
- 2.-Evaluar el comportamiento y observar si es el comportamiento deseado.

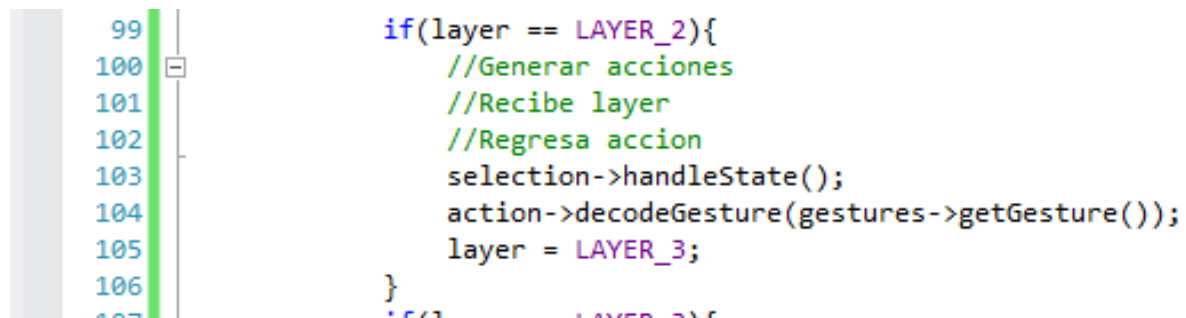
Responsable de ejecución: Arquitecto

Resultados Esperados

Se espera que a partir de una gestura dada se obtenga una acción concreta.

Resultados Obtenidos

Para la capa selection se obtuvo un resultado satisfactorio al aislar por completo la capa y observar su comportamiento a detalle, se tienen todas las acciones a partir de la entrada directa de todas las gesturas, es importante destacar que no todas las gesturas tienen una acción dada y por tanto esto no entorpece el funcionamiento del sistema, el único comportamiento que tiene es el de no hacer nada.



```
99  
100  
101  
102  
103  
104  
105  
106  
107  
  
if(layer == LAYER_2){  
    //Generar acciones  
    //Recibe layer  
    //Regresa accion  
    selection->handleState();  
    action->decodeGesture(gestures->getGesture());  
    layer = LAYER_3;  
}
```

Prueba 4.3 Se evalúa la capa translation para la acción del robot

Proceso

- 1.-Segmentar el sistema a la capa translation.
- 2.-Evaluar el comportamiento y observar si es el comportamiento deseado.

Responsable de ejecución: Arquitecto

Resultados Esperados

Se espera que a partir de una acción dada se obtenga una acción concreta contenida en una configuración de robot.

Resultados Obtenidos

Para la capa translation se obtuvo un resultado satisfactorio al aislar por completo la capa y observar su comportamiento a detalle, la transición de una acción general a una acción concreta dependiendo una configuración específica de robot es correcta ya que estas se limitan a pre configuraciones, en caso de encontrar algo desconocido el sistema no falla y continua su ejecución.

```
if(layer == LAYER_3){  
    //Se obtiene un tipo de robot  
    //Recibe accion  
    //Regresa instrucciones  
    traduction->handleState();  
    robbie->setActionRobot(action->getAction());  
    robbie->decodeAction();  
    robbie->getInstructions();  
    layer = LAYER_4;  
}
```

Prueba 4.4 Se evalúa la capa transformation para las instrucciones del robot

Proceso

- 1.-Segmentar el sistema a la capa transformation.
- 2.-Evaluar el comportamiento y observar si es el comportamiento deseado.

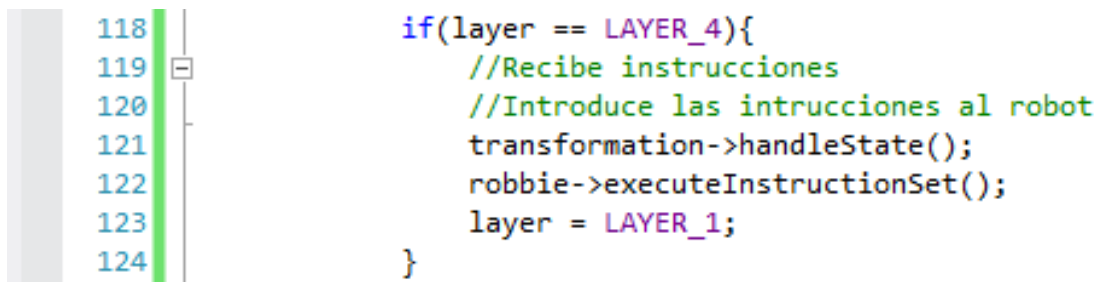
Responsable de ejecución: Arquitecto

Resultados Esperados

Se espera que a partir de una acción concreta del robot se genere un conjunto de instrucciones específicas del robot configurado.

Resultados Obtenidos

Para la capa transformation se ejecutó correctamente la prueba y el aislamiento de la capa, el conjunto de instrucciones generado es de acuerdo a los parámetros configurados siguiendo la capacidad que cuentan los motores y al nivel necesario. En esta prueba es importante destacar que se observa más claramente la oportunidad de agregar un patrón de diseño para la salida de información.



```
118  
119  
120  
121  
122  
123  
124  
  
if(layer == LAYER_4){  
    //Recibe instrucciones  
    //Introduce las intrucciones al robot  
    transformation->handleState();  
    robbie->executeInstructionSet();  
    layer = LAYER_1;  
}
```

Prueba 4.5 Se evalúa la capa de implementación para el Robot

Proceso

- 1.-Segmentar el sistema a la capa implementación.
- 2.-Evaluar el comportamiento y observar si es el comportamiento deseado.

Responsable de ejecución: Arquitecto

Resultados Esperados

Se espera que a partir de un conjunto de instrucciones el robot ejecute las mismas de acuerdo a su librería de ejecución en puerto serial.

Resultados Obtenidos

Para la capa implementación se ejecutó correctamente la prueba y el aislamiento de la capa, se observa que la aplicación del sistema puede incrementarse dando apertura a más tipos de salida de instrucciones adecuando según el tipo de robot.

```
296 void Robot::getInstructions(){
297     instructions.clear();
298     for(int indx = 0; indx < children.size(); indx++){
299         children[indx]->getInstruction(&instructions);
300     }
301
302     std::cout << std::endl << "Instructions:" << std::endl;
303     for(int i = 0; i < instructions.size(); i++){
304         std::cout << i << " Motor:" << instructions[i].id_motor << " " << instructions[i].value << " " << instructions[i].speed << std
305     }
306 }
307
308 bool Robot::executeInstructionSet(){
309     if(instructions.data() != NULL){
310         Instruction *instruction = new Instruction();
311         int res = -1;
312
313         if(type == MOBILE){
314             res = instruction->moveWithSpeed(instructions);
315         }else if(type == HUMANOID){
316             res = instruction->moveToPosition(instructions);
317         }
318
319         if(res == -1){
320             return false;
321         }else{
322             return true;
323         }
324     }else{
325         return false;
326     }
327 }
328 }
```


Prueba 5.1 Se agrega una capa evaluativa al inicio del proceso de capas

Proceso

- 1.-Abrir el entorno de desarrollo del sistema.
- 2.-Colocar el entorno de desarrollo al inicio de la sección de capas.
- 3.-Hacer que el usuario cree una nueva capa y la agregue de manera intuitiva al sistema actual.

Responsable de ejecución: Relacionados

Resultados Esperados

Se espera que un usuario pueda agregar una capa al sistema de manera intuitiva y sencilla enfocándose más en lo que realiza la capa que en agregarla como tal. Además se espera que la nueva capa se acople de manera natural al sistema mientras cumpla con las condiciones que propone el sistema.

Resultados Obtenidos

La prueba presentó resultados exitosos cuando que un usuario puede crear de manera sencilla una capa permitiendo la extensibilidad del sistema en cuanto a la arquitectura general al inicio de la misma.

```

161         if(layer == LAYER_0){
162             //printf("LAYER 1\n");
163             //cout << "SET THE FIRST INSTRUCTION" << endl;
164             NuiSkeletonGetNextFrame(0, &ourframe);
165             for(int i=0; i<6; i++){
166                 if(ourframe.SkeletonData[i].eTrackingState == NUI_SKELETON_TRACKED){
167
168                     gestures->setGesture(
169                         gestures->decodeGesture(
170                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_RIGHT].x,
171                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_RIGHT].y,
172                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_RIGHT].z,
173                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_LEFT].x,
174                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_LEFT].y,
175                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_LEFT].z,
176                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_SHOULDER_CENTER].x,
177                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_SHOULDER_CENTER].y,
178                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_SHOULDER_CENTER].z
179                         )
180                     );
181                     switch(gestures->getGesture()){
182                         case EQUAL_RIGHT_HANDS:
183                             //cout << "EQUAL RIGHT HANDS" << endl;
184                             layer = LAYER_1;
185                             break;
186                         case EQUAL_LEFT_HANDS:
187                             //cout << "EQUAL LEFT HANDS" << endl;
188                             layer = LAYER_1;
189                             break;
190                     }
191                 }
192             }
193         }
194         if(layer == LAYER_1){

```

Prueba 5.2 Se agrega una capa evaluativa al medio del proceso de capas

Proceso

- 1.-Abrir el entorno de desarrollo del sistema.
- 2.-Colocar el entorno de desarrollo a la mitad de la sección de capas.
- 3.-Hacer que el usuario cree una nueva capa y la agregue de manera intuitiva al sistema actual.

Responsable de ejecución: Relacionados

Resultados Esperados

Se espera que un usuario pueda agregar una capa al sistema de manera intuitiva y sencilla enfocándose más en lo que realiza la capa que en agregarla como tal. Además se espera que la nueva capa se acople de manera natural al sistema mientras cumpla con las condiciones que propone el sistema.

Resultados Obtenidos

La prueba presentó resultados exitosos cuando que un usuario puede crear de manera sencilla una capa permitiendo la extensibilidad del sistema en cuanto a la arquitectura general a la mitad de la misma.

```
161         if(layer == LAYER_0){
162             //printf("LAYER 1\n");
163             //cout << "SET THE FIRST INSTRUCTION" << endl;
164             NuiSkeletonGetNextFrame(0, &ourframe);
165             for(int i=0; i<6; i++){
166                 if(ourframe.SkeletonData[i].eTrackingState == NUI_SKELETON_TRACKED){
167
168                     gestures->setGesture(
169                         gestures->decodeGesture(
170                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_RIGHT].x,
171                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_RIGHT].y,
172                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_RIGHT].z,
173                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_LEFT].x,
174                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_LEFT].y,
175                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_LEFT].z,
176                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_SHOULDER_CENTER].x,
177                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_SHOULDER_CENTER].y,
178                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_SHOULDER_CENTER].z
179                         )
180                     );
181                     switch(gestures->getGesture()){
182                         case EQUAL_RIGHT_HANDS:
183                             //cout << "EQUAL RIGHT HANDS" << endl;
184                             layer = LAYER_1;
185                             break;
186                         case EQUAL_LEFT_HANDS:
187                             //cout << "EQUAL LEFT HANDS" << endl;
188                             layer = LAYER_1;
189                             break;
190                     }
191                 }
192             }
193         }
194         if(layer == LAYER_3){
195             //printf("LAYER 1\n");
196         }
197     }
```

Prueba 5.3 Se agrega una capa evaluativa al final del proceso de capas

Proceso

- 1.-Abrir el entorno de desarrollo del sistema.
- 2.-Colocar el entorno de desarrollo al final de la sección de capas.
- 3.-Hacer que el usuario cree una nueva capa y la agregue de manera intuitiva al sistema actual.

Responsable de ejecución: Relacionados

Resultados Esperados

Se espera que un usuario pueda agregar una capa al sistema de manera intuitiva y sencilla enfocándose más en lo que realiza la capa que en agregarla como tal. Además se espera que la nueva capa se acople de manera natural al sistema mientras cumpla con las condiciones que propone el sistema.

Resultados Obtenidos

La prueba presentó resultados exitosos cuando que un usuario puede crear de manera sencilla una capa permitiendo la extensibilidad del sistema en cuanto a la arquitectura general al final de la misma.

```

161         if(layer == LAYER_4){
162             //Recibe instrucciones
163             //Introduce las instrucciones al robot
164             //transformation->handleState();
165             robbie->executeInstructionSet();
166             layer = LAYER_1;
167         }
168         if(layer == LAYER_0){
169             //printf("LAYER 1\n");
170             //cout << "SET THE FIRST INSTRUCTION" << endl;
171             NuiSkeletonGetNextFrame(0, &ourframe);
172             for(int i=0; i<6; i++){
173                 if(ourframe.SkeletonData[i].eTrackingState == NUI_SKELETON_TRACKED){
174
175                     gestures->setGesture(
176                         gestures->decodeGesture(
177                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_RIGHT].x,
178                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_RIGHT].y,
179                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_RIGHT].z,
180                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_LEFT].x,
181                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_LEFT].y,
182                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_HAND_LEFT].z,
183                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_SHOULDER_CENTER].x,
184                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_SHOULDER_CENTER].y,
185                             ourframe.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_SHOULDER_CENTER].z
186                         )
187                     );
188                     switch(gestures->getGesture()){
189                         case EQUAL_RIGHT_HANDS:
190                             //cout << "EQUAL RIGHT HANDS" << endl;
191                             layer = LAYER_1;
192                             break;
193                         case EQUAL_LEFT_HANDS:
194                             //cout << "EQUAL LEFT HANDS" << endl;
195                             layer = LAYER_1;
196                     }
197                 }
198             }
199         }
200     }
201 }

```

Prueba 5.4 Se agrega una sub capa evaluativa al inicio de un módulo del sistema

Proceso

- 1.-Abrir el entorno de desarrollo del sistema.
- 2.-Colocar el entorno de desarrollo dentro del módulo de gesturas.
- 3.-Hacer que el usuario cree una nueva gestura.

Responsable de ejecución: Relacionados

Resultados Esperados

Se espera que el usuario pueda agregar una nueva gestura de manera intuitiva y de manera simple, siendo natural la agregación y que no se comprometa el sistema.

Resultados Obtenidos

La prueba presentó resultados exitosos, la agregación de nuevas gesturas al sistema es de manera simple e intuitiva para los usuarios.

```
30 int Gestures::decodeGesture(float rX, float rY, float rZ, float lX, float lY, float lZ, float sCX, float sCY, float sCZ){
31
32
33     //Set hand to undefined
34     handLeftX = handLeftY = handLeftZ = handRightX = handRightY = handRightZ = HAND_UNDEFINED;
35
36
37     if(rX > sCX && rX > 0.4 ){
38         handRightX = HAND_RIGHT;
39     }
40     if(lX < sCX && lX < -0.4){
41         handLeftX = HAND_LEFT;
42     }
43     if(lX > sCX){
44         handLeftX = HAND_RIGHT;
45     }
46     if(rX < sCX){
47         handRightX = HAND_LEFT;
48     }
49     if(rY > 0.5){
50         handRightY = HAND_UP;
51     }
52     if(lY > 0.5){
53         handLeftY = HAND_UP;
54     }
55     if(rY < -0.2){
56         handRightY = HAND_DOWN;
57     }
58     if(lY < -0.2){
59         handLeftY = HAND_DOWN;
60     }
61 }
```

Prueba 5.5 Se agrega una sub capa evaluativa al medio de un módulo del sistema

Proceso

- 1.-Abrir el entorno de desarrollo del sistema.
- 2.-Colocar el entorno de desarrollo dentro del módulo de acciones.
- 3.-Hacer que el usuario cree una nueva acción.

Responsable de ejecución: Relacionados

Resultados Esperados

Se espera que el usuario pueda agregar una nueva acción al sistema de manera intuitiva y natural, sin comprometer el sistema ni su funcionalidad.

Resultados Obtenidos

La prueba presentó resultados exitosos, la agregación de nuevas acciones al sistema es de manera simple e intuitiva para los usuarios.

```
30 int Gestures::decodeGesture(float rX, float rY, float rZ, float lX, float lY, float lZ, float sCX, float sCY, float sCZ){
31
32
33     //Set hand to undefined
34     handLeftX = handLeftY = handLeftZ = handRightX = handRightY = handRightZ = HAND_UNDEFINED;
35
36
37     if(rX > sCX && rX > 0.4 ){
38         handRightX = HAND_RIGHT;
39     }
40     if(lX < sCX && lX < -0.4){
41         handLeftX = HAND_LEFT;
42     }
43     if(lX > sCX){
44         handLeftX = HAND_RIGHT;
45     }
46     if(rX < sCX){
47         handRightX = HAND_LEFT;
48     }
49     if(rY > 0.5){
50         handRightY = HAND_UP;
51     }
52     if(lY > 0.5){
53         handLeftY = HAND_UP;
54     }
55     if(rY < -0.2){
56         handRightY = HAND_DOWN;
57     }
58     if(lY < -0.2){
59         handLeftY = HAND_DOWN;
60     }
61 }
```

Prueba 5.6 Se agrega una sub capa evaluativa al final de un módulo del sistema

Proceso

- 1.-Abrir el entorno de desarrollo del sistema.
- 2.-Colocar el entorno de desarrollo dentro del módulo acciones de robots.
- 3.-Hacer que el usuario cree una nueva acción del robot.

Responsable de ejecución: Relacionados

Resultados Esperados

Se espera que el usuario pueda agregar una nueva acción al robot en el sistema de manera intuitiva y natural, sin comprometer el sistema ni su funcionalidad.

Resultados Obtenidos

La prueba presentó resultados exitosos, la agregación de nuevas acciones robots al sistema es de manera simple e intuitiva para los usuarios.

```
30 int Gestures::decodeGesture(float rX, float rY, float rZ, float lX, float lY, float lZ, float sCX, float sCY, float sCZ){
31
32
33     //Set hand to undefined
34     handLeftX = handLeftY = handLeftZ = handRightX = handRightY = handRightZ = HAND_UNDEFINED;
35
36
37     if(rX > sCX && rX > 0.4 ){
38         handRightX = HAND_RIGHT;
39     }
40     if(lX < sCX && lX < -0.4){
41         handLeftX = HAND_LEFT;
42     }
43     if(lX > sCX){
44         handLeftX = HAND_RIGHT;
45     }
46     if(rX < sCX){
47         handRightX = HAND_LEFT;
48     }
49     if(rY > 0.5){
50         handRightY = HAND_UP;
51     }
52     if(lY > 0.5){
53         handLeftY = HAND_UP;
54     }
55     if(rY < -0.2){
56         handRightY = HAND_DOWN;
57     }
58     if(lY < -0.2){
59         handLeftY = HAND_DOWN;
60     }
61 }
```

Prueba 5.7 Se agrega una rutina nueva para una acción dada en el sistema para algún tipo de robot

Proceso

- 1.-Abrir el entorno de desarrollo del sistema.
- 2.-Colocar el entorno de desarrollo dentro del módulo de robots.
- 3.-Hacer que el usuario cree un nuevo tipo de robot.

Responsable de ejecución: Relacionados

Resultados Esperados

Se espera que el usuario pueda agregar tipo de robot en el sistema de manera intuitiva y natural, sin comprometer el sistema ni su funcionalidad.

Resultados Obtenidos

La prueba presentó resultados exitosos, la agregación de nuevos tipos de robots al sistema es de manera simple e intuitiva para los usuarios.

```

197         break;
198     case HUMANOID:
199         Component *right_arm;
200         Component *left_arm;
201         switch (action->getAction())
202         {
203             case CENTER_SELF:
204
205                 right_arm = searchComponent("Right Arm");
206                 left_arm = searchComponent("Left Arm");
207                 if(right_arm != NULL && left_arm != NULL){
208                     right_arm->stablishValueLeaf("motor3", 512, 0);
209                     right_arm->stablishValueLeaf("motor5", 512, 0);
210                     left_arm->stablishValueLeaf("motor4", 512, 0);
211                     left_arm->stablishValueLeaf("motor6", 512, 0);
212                 }
213                 //printHierarchy();
214                 break;
215             case MOVEMENT_UP:
216
217                 right_arm = searchComponent("Right Arm");
218                 left_arm = searchComponent("Left Arm");
219                 if(right_arm != NULL && left_arm != NULL){
220                     right_arm->stablishValueLeaf("motor3", 650, 0);
221                     right_arm->stablishValueLeaf("motor5", 512, 0);
222                     left_arm->stablishValueLeaf("motor4", 370, 0);
223                     left_arm->stablishValueLeaf("motor6", 512, 0);
224                 }
225                 //printHierarchy();
226                 break;
227             case MOVEMENT_DOWN:
228
229                 right_arm = searchComponent("Right Arm");
230                 left_arm = searchComponent("Left Arm");
231                 if(right_arm != NULL && left_arm != NULL){

```

Prueba 5.8 Se modifica una rutina existente para una acción dada de un robot

Proceso

- 1.-Abrir el entorno de desarrollo del sistema.
- 2.-Colocar el entorno de desarrollo dentro del módulo de rutinas del robot.
- 3.-Hacer que el usuario cree una nueva rutina para un tipo de robot.

Responsable de ejecución: Relacionados

Resultados Esperados

Se espera que el usuario pueda agregar una acción a un tipo de robot en el sistema de manera intuitiva y natural, sin comprometer el sistema ni su funcionalidad.

Resultados Obtenidos

La prueba presentó resultados exitosos, la agregación de nuevas rutinas a los robots del sistema es de manera simple e intuitiva para los usuarios.

```

197         break;
198     case HUMANOID:
199         Component *right_arm;
200         Component *left_arm;
201         switch (action->getAction())
202         {
203             case CENTER_SELF:
204
205                 right_arm = searchComponent("Right Arm");
206                 left_arm = searchComponent("Left Arm");
207                 if(right_arm != NULL && left_arm != NULL){
208                     right_arm->stablishValueLeaf("motor3", 512, 0);
209                     right_arm->stablishValueLeaf("motor5", 512, 0);
210                     left_arm->stablishValueLeaf("motor4", 512, 0);
211                     left_arm->stablishValueLeaf("motor6", 512, 0);
212                 }
213                 //printHierarchy();
214                 break;
215             case MOVEMENT_UP:
216
217                 right_arm = searchComponent("Right Arm");
218                 left_arm = searchComponent("Left Arm");
219                 if(right_arm != NULL && left_arm != NULL){
220                     right_arm->stablishValueLeaf("motor3", 650, 0);
221                     right_arm->stablishValueLeaf("motor5", 512, 0);
222                     left_arm->stablishValueLeaf("motor4", 370, 0);
223                     left_arm->stablishValueLeaf("motor6", 512, 0);
224                 }
225                 //printHierarchy();
226                 break;
227             case MOVEMENT_DOWN:
228
229                 right_arm = searchComponent("Right Arm");
230                 left_arm = searchComponent("Left Arm");
231                 if(right_arm != NULL && left_arm != NULL){

```


Prueba 5.9 Se elimina una rutina existente de un robot para una acción dada

Proceso

- 1.-Abrir el entorno de desarrollo del sistema.
- 2.-Colocar el entorno de desarrollo dentro del módulo de rutinas del robot.
- 3.-Hacer que el elimine una rutina aleatoria de algún tipo de robot configurado en el sistema.
- 4.-Ejecutar el sistema.
- 5.-Seleccionar el tipo de robot a utilizar.
- 6.-Introducir una gestura al sistema.

Responsable de ejecución: Relacionados

Resultados Esperados

Se espera que el sistema soporte la eliminación repentina de una rutina sin comprometer la funcionalidad y estabilidad de todo el sistema.

Resultados Obtenidos

La prueba se ejecutó de manera exitosa dejando ver que el sistema tiene soporte a reemplazabilidad de componentes del mismo.

```

197 break;
198 case HUMANOID:
199     Component *right_arm;
200     Component *left_arm;
201     switch (action->getAction())
202     {
203         case CENTER_SELF:
204
205             right_arm = searchComponent("Right Arm");
206             left_arm = searchComponent("Left Arm");
207             if(right_arm != NULL && left_arm != NULL){
208                 right_arm->stabilshValueLeaf("motor3", 512, 0);
209                 right_arm->stabilshValueLeaf("motors", 512, 0);
210                 left_arm->stabilshValueLeaf("motor4", 512, 0);
211                 left_arm->stabilshValueLeaf("motor6", 512, 0);
212             }
213             //printHierarchy();
214             break;
215         case MOVEMENT_UP:
216
217             right_arm = searchComponent("Right Arm");
218             left_arm = searchComponent("Left Arm");
219             if(right_arm != NULL && left_arm != NULL){
220                 right_arm->stabilshValueLeaf("motor3", 650, 0);
221                 right_arm->stabilshValueLeaf("motors", 512, 0);
222                 left_arm->stabilshValueLeaf("motor4", 370, 0);
223                 left_arm->stabilshValueLeaf("motor6", 512, 0);
224             }
225             //printHierarchy();
226             break;
227         case MOVEMENT_DOWN:
228
229             right_arm = searchComponent("Right Arm");
230             left_arm = searchComponent("Left Arm");
231
232             right_arm = searchComponent("Right Arm");
233             left_arm = searchComponent("Left Arm");
234             if(right_arm != NULL && left_arm != NULL){
235                 right_arm->stabilshValueLeaf("motor3", 512, 0);
236                 right_arm->stabilshValueLeaf("motors", 512, 0);
237                 left_arm->stabilshValueLeaf("motor4", 512, 0);
238                 left_arm->stabilshValueLeaf("motor6", 512, 0);
239             }
240             //printHierarchy();
241             break;
242         case MOVEMENT_LEFT:
243
244             right_arm = searchComponent("Right Arm");
245             left_arm = searchComponent("Left Arm");
246             if(right_arm != NULL && left_arm != NULL){
247                 right_arm->stabilshValueLeaf("motor3", 370, 0);
248                 right_arm->stabilshValueLeaf("motors", 512, 0);
249                 left_arm->stabilshValueLeaf("motor4", 650, 0);
250                 left_arm->stabilshValueLeaf("motor6", 512, 0);
251             }
252             //printHierarchy();
253             break;
254         case MOVEMENT_RIGHT:
255
256             right_arm = searchComponent("Right Arm");
257             left_arm = searchComponent("Left Arm");
258             if(right_arm != NULL && left_arm != NULL){
259                 right_arm->stabilshValueLeaf("motor3", 512, 0);
260                 right_arm->stabilshValueLeaf("motors", 512, 0);
261                 left_arm->stabilshValueLeaf("motor4", 512, 0);
262                 left_arm->stabilshValueLeaf("motor6", 512, 0);
263             }
264             //printHierarchy();
265             break;
266     }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
8
```

Conclusiones de las evaluaciones

A partir de las evaluaciones generadas se obtuvo que el sistema cumpla con los atributos de calidad planteados en un inicio. Estos atributos definidos en la parte inicial de este documento permiten que el sistema tenga flexibilidad ante adversidades, crecimiento y mantenibilidad principalmente. Además las evaluaciones permitieron observar la falta de patrones de diseño que después se agregó e implementó permitiendo una mejor flexibilidad sobre todo para la entrada y salida de la información.

3. Conclusiones Generales

Este proyecto permitió hacer una investigación sobre diferentes temas: arquitectura de software, atributos de calidad, metodología de evaluación, robótica, interfaces naturales, sistemas embebidos y microcontroladores. Cada uno de estos temas en sí mismo permitió desarrollar capacidades propias del tema logrando como resultado un sistema simple, eficaz y extensible de robótica. Es probable que el sistema tenga futuro de investigación para la extensibilidad de nuevos módulos o nuevos modelos de comunicación con otro tipo de robots así como poder volverse un intermediario entre dispositivos de conexión como pueden ser otras computadoras, dispositivos móviles o nuevas tecnologías que surjan. Uno de los principales fuertes del proyecto es el uso de patrones de diseño para permitir guardar la jerarquía de la arquitectura de software por capas con la cual está montado el sistema. El uso de estos patrones permite mejorar la creación de nuevas formas en que el sistema interactúa así como también define la manera en que el mismo sistema se comporta. Sin el uso de estos patrones el sistema hubiera sido posible, sin embargo no hubiera tenido la suficiente flexibilidad como hasta ahora la tiene.

4. Bibliografía

ACEVES, A. (2010). SERVOMOTORES DYNAMIXEL. SEMINARIO DEL GRUPO DE INVESTIGACIÓN DE HUMANOIDES.

FREEMAN, E. (2004). HEAD FIRST DESIGN PATTERNS. O'REILLY. ESTADOS UNIDOS.

JOYANES, L. (2011). PROGRAMACIÓN EN C++, JAVA Y UML. MC GRAW HILL. ESPAÑA.

CORTES, J. (2011). BIOLOID C++ TUTORIAL. VISITADO EL 6 DE SEPTIEMBRE, 2013 DE [HTTP://SOFTWARESOULS.COM/SOFTWARESOULS/SERIES/BIOLOID-C-TUTORIAL/](http://SOFTWARESOULS.COM/SOFTWARESOULS/SERIES/BIOLOID-C-TUTORIAL/)

CORTES, J. (2011). PRACTICAL C++ PROGRAMMING TUTORIAL FOR BIOLOID ROBOTS. VISITADO EL 8 DE SEPTIEMBRE, 2013 DE [HTTP://SOFTWARESOULS.COM/SOFTWARESOULS/2011/11/23/PRACTICAL-C-PROGRAMMING-TUTORIAL-FOR-BIOLOID-ROBOTS/](http://SOFTWARESOULS.COM/SOFTWARESOULS/2011/11/23/PRACTICAL-C-PROGRAMMING-TUTORIAL-FOR-BIOLOID-ROBOTS/)

CORTES, J. (2011). READING DYNAMIXEL AX-12+ POSITION. VISITADO EL 8 DE SEPTIEMBRE, 2013 DE [HTTP://SOFTWARESOULS.COM/SOFTWARESOULS/2011/11/23/READING-DYNAMIXEL-AX12-POSITION/](http://SOFTWARESOULS.COM/SOFTWARESOULS/2011/11/23/READING-DYNAMIXEL-AX12-POSITION/)

CORTES, J. (2011). WRITING DYNAMIXEL AX-12+ POSITION. VISITADO EL 14 DE SEPTIEMBRE, 2013 DE [HTTP://SOFTWARESOULS.COM/SOFTWARESOULS/2011/12/11/WRITING-DYNAMIXEL-AX-12-POSITION/](http://SOFTWARESOULS.COM/SOFTWARESOULS/2011/12/11/WRITING-DYNAMIXEL-AX-12-POSITION/)

CORTES, J. (2011). LINUX C++ DYNAMIXEL READING AND WRITING EXAMPLE. VISITADO EL 14 DE SEPTIEMBRE, 2013 DE [HTTP://SOFTWARESOULS.COM/SOFTWARESOULS/2012/03/05/LINUX-C-DYNAMIXEL-READING-AND-WRITING-EXAMPLE/](http://SOFTWARESOULS.COM/SOFTWARESOULS/2012/03/05/LINUX-C-DYNAMIXEL-READING-AND-WRITING-EXAMPLE/)

RACHAEL, M (ENERO 14, 2013). KINECT TUTORIAL (C++). VISITADA EL 18 DE SEPTIEMBRE, 2013 DE [HTTP://ELECTRONICLUNCH.WORDPRESS.COM/2013/01/14/KINECT-TUTORIAL-C/](http://ELECTRONICLUNCH.WORDPRESS.COM/2013/01/14/KINECT-TUTORIAL-C/)

ROBOTIS (2010). ROBOTIS E-MANUAL V1.15.00. VISITADO EL 14 DE SEPTIEMBRE, 2013 DE [HTTP://SUPPORT.ROBOTIS.COM/EN/](http://SUPPORT.ROBOTIS.COM/EN/)

ROBOTIS (2010). ROBOTIS E-MANUAL V1.15.00 AX-12/ AX-12+ AX-12A. VISITADO EL 14 DE SEPTIEMBRE, 2013 DE [HTTP://SUPPORT.ROBOTIS.COM/EN/PRODUCT/DYNAMIXEL/AX_SERIES/DXL_AX_ACTUATOR.HTM#ACTUATOR_ADDRESS_2E](http://SUPPORT.ROBOTIS.COM/EN/PRODUCT/DYNAMIXEL/AX_SERIES/DXL_AX_ACTUATOR.HTM#ACTUATOR_ADDRESS_2E)