I. El Proyecto

I.1. Identidad del Proyecto

I.1.1. Título del Proyecto

Control del Brazo Robótico Scorbot er4u Mediante Matlab.

I.1.2. Carrera

Ingeniería Informática.

I.1.3. Facultad

Facultad de Ciencias y Tecnología.

I.1.4. Duración del Proyecto

8 meses.

I.1.5. Área/Línea de Investigación Priorizada

Investigación y Robótica.

I.2. Personal vinculado al Proyecto

I.2.1. Director del Proyecto

Univ. Daniel Osmar Subelza Flores.

Tel.: 466-46852.

Cel.: 72976542.

Correo: daniels_0076@hotmail.com

I.2.2. Equipo de Trabajo de: Empresa/Institución/Organizaciones participantes/cooperantes

Universidad Autónoma Juan Misael Saracho, Facultad de Ciencias y Tecnología, Ingeniería Informática, Laboratorio de Robótica.

I.2.3. Actividades previstas para los integrantes del equipo de investigación

Director, Investigador.

Encargado de realizar:

- Planificación y control del proyecto según cronograma.
- Mantenerse enfocado en los objetivos.

  Supervisar el desarrollo del cronograma del proyecto de acuerdo al cronograma de actividades y desarrollar todas las actividades en el proyecto para que estas puedan cumplirse en el tiempo planificado en el proyecto para así poder llegar a cumplir todas las metas y objetivos trazados por el proyecto.

- Determinar la especificación y validación de requerimientos de acuerdo a las necesidades del operador.
- Elaboración del análisis, diseño, implementación, programación y desarrollo del proyecto.
- Diseño de estructura del sistema, diagramas UML.
- Diseño de las interfaces de usuario del sistema de forma que el sistema para que su uso sea intuitivo, amigable y este de acuerdo con las actividades que realizara dicho sistema.
- Programación del sistema de acuerdo al análisis y diseño establecidos.
- Preparación de pruebas funcionales del sistema aplicando la ingeniería de software.
- Elaboración de la documentación del proyecto.
- Elaborar modelos de componentes y despliegue.
- Diseño y elaboración de las guías del curso de capacitación del sistema.
- Notificación a la población universitaria involucrada para la capacitación del manejo y automatizado, propiamente de la

Facultad de Ciencias y Tecnología de la Carrera de Ingeniería Informática que cursen entre 4to y 5to año.

- Capacitar al personal docente involucrado en el uso del sistema.

Y también será responsable del seguimiento, estudio de una solución de código abierto para el brazo robotico scorbot er4u. Para este fin se estudiara la accesibilidad, compatibilidad y comunicación entre matlab y el brazo robot scorbot er 4u

Analista.

Como analista será el responsable de la recopilación de información, clasificación e interpretación para el desarrollo del estudio de la comunicación entre el programa matemático matlab y el brazo robot.

Programador.

Como programador tendrá que realizar la interpretación plasmada en la información hecha por el analista, pasándola a un lenguaje entendible para la computadora, es este caso, pasando a lenguaje C entendible por matlab.

Asesor.

Hará el seguimiento corroborando que la aplicación desarrollada para el robot scorbot er4u se aplique de una forma adecuada.

I.3. Descripción del Proyecto

I.3.1. Introducción

Durante la evolución de las maquinas industriales, el hombre se ha sentido fascinado por las maquinarias y dispositivos capaces de replicar actividades y movimientos de otros seres vivos ya sean humanos, animales o insectos. De esta manera, hoy tenemos en

muchas de las industrias del mundo, robots diseñados con particularidades humanos, como es el caso de los brazos robots.

Estos equipos, por darle un calificativo, que en su tiempo fue un elemento de inspiración de la ciencia ficción, hoy es una realidad empleada en los procesos industriales con toda una infraestructura compleja.

Al involucrarse en la utilización de robots, uno de los aspectos importantes que hay que tomar en cuenta es "la ley de la robótica", expresada de la siguiente forma:

- Un robot no puede perjudicar a un ser humano, ni con su inacción permitir que un ser humano sufra daño.
- Un robot debe obedecer las órdenes recibidas de un ser humano, excepto si tales órdenes entran en conflicto con la primera ley.
- Un robot debe proteger su propia existencia mientras tal protección no entre en conflicto con la primera y segunda ley.

Por otro lado, si se debe definir el concepto de robot industrial, la federación internación de robótica (FIR) lo determina de la siguiente manera: "por robot industrial de manipulación, se entiende a una máquina de manipulación automática, reprogramable y multifuncional con tres o más ejes que pueden posicionar y orientar materias, piezas, herramientas o dispositivos especiales para la ejecución de trabajos diversos en las diferentes etapas de la producción industrial, ya sea en una posición fija o de movimiento[1]".

I.3.2. Antecedentes

I.3.2.1. Antecedentes en robótica industrial

[1]brazo, G. d. (s.f.). Obtenido de http://www.dspace.espol.edu.ec/bitstream/123456789/6906/1/Guía%20de%20Laboratorio%20del%20Brazo%20Robot%20Scorbot%20ER%204u.pdf

Aplicaciones para brazos robots en nuestro medio (Tarija-Bolivia), o la automatización mecánica en labores manuales que exijan exactitud, fuerza y rapidez con un bajo margen de error es poca, siendo empresas tales como embotelladoras[2] con un sistema de embotellado mecánico automatizado, universidades que aplican con propósitos académicos, conferencias sobre robótica industrial con especialistas extranjeros[3], que solo queda en teoría los únicos expositores de esta grandiosa rama de la robótica industrial, además de casos especiales de gente en Tarija que tiene robots para la limpieza de sus hogares.

I.3.2.2. Antecedentes de proyectos anteriores[4]

Para este proyecto no se encontró precedentes que involucren la utilización de un entorno visual y simulación para el control del brazo robot por medio de matlab, ya sea en tesis presentadas en otras gestiones o web.

I.3.3. Descripción y Fundamentación del Proyecto (qué y por qué)

El proyecto "Control del Brazo Robótico Scorbot er4u mediante Matlab" tiene la finalidad de hacer uso de herramientas, como ser el Brazo Robótico Scorbot er4u para que la población universitaria y docente de la carrera de Ingeniería Informática pueda continuar con la investigación y realizar otros aportes en el campo de la robótica teniendo como precedente a esta aplicación.

---

[2] Coca Cola, Cascada.

[3] Encontrado en:
U.C.B. (Agosto de 2011). Obtenido de
http://lpz.ucb.edu.bo/Forms/Noticias/NoticiasUCB.aspx?NSNoticia=74441

[4] Control del Brazo Robot mediante Matlab

Otra finalidad con este proyecto es la de desarrollar un software en MatLab que pueda controlar al brazo robótico, si bien el software con el que viene el scorbot realiza estas operaciones, pero sin embargo su código es cerrado, con la presentación del software a desarrollar, se quiere dejar un precedente para futuras investigaciones, donde podrán modificar el código del software a desarrollar tanto como sea necesario.

El proyecto ayudará de alguna manera en el entendimiento más concreto de la funcionalidad del brazo robótico.

I.3.3.1. Justificación tecnológica

En nuestro medio se cuenta por así decirlo con la tecnología adecuada que se requiere para la puesta en marcha de este proyecto. Ya que los requerimientos tanto hardware como software son accesibles. En el área de las ciencias y tecnologías es importante incentivar las distintas modalidades de la informática, en este caso se presentó la posibilidad de hacerlo y el campo será en la robótica.

I.3.3.2. Justificación social

La universidad debe jugar un papel importante en la innovación tecnológica en la sociedad, dando respuestas a las necesidades de la misma, ofreciendo herramientas que agilicen y simplifiquen los procedimientos previos a los cambios tecnológicos que se quiera implementar.

I.3.3.3. Justificación académica

En el campo académico observamos que la robótica se deja un poco de lado, porque al no haber tantas industrias que sean merecedoras para este cambio, no se da un gran énfasis, como ser una planta de ensamblaje de automóviles, armado de circuitos integrados a la altura de procesadores AMD o Intel.

I.3.4. Cuadro de Involucrados

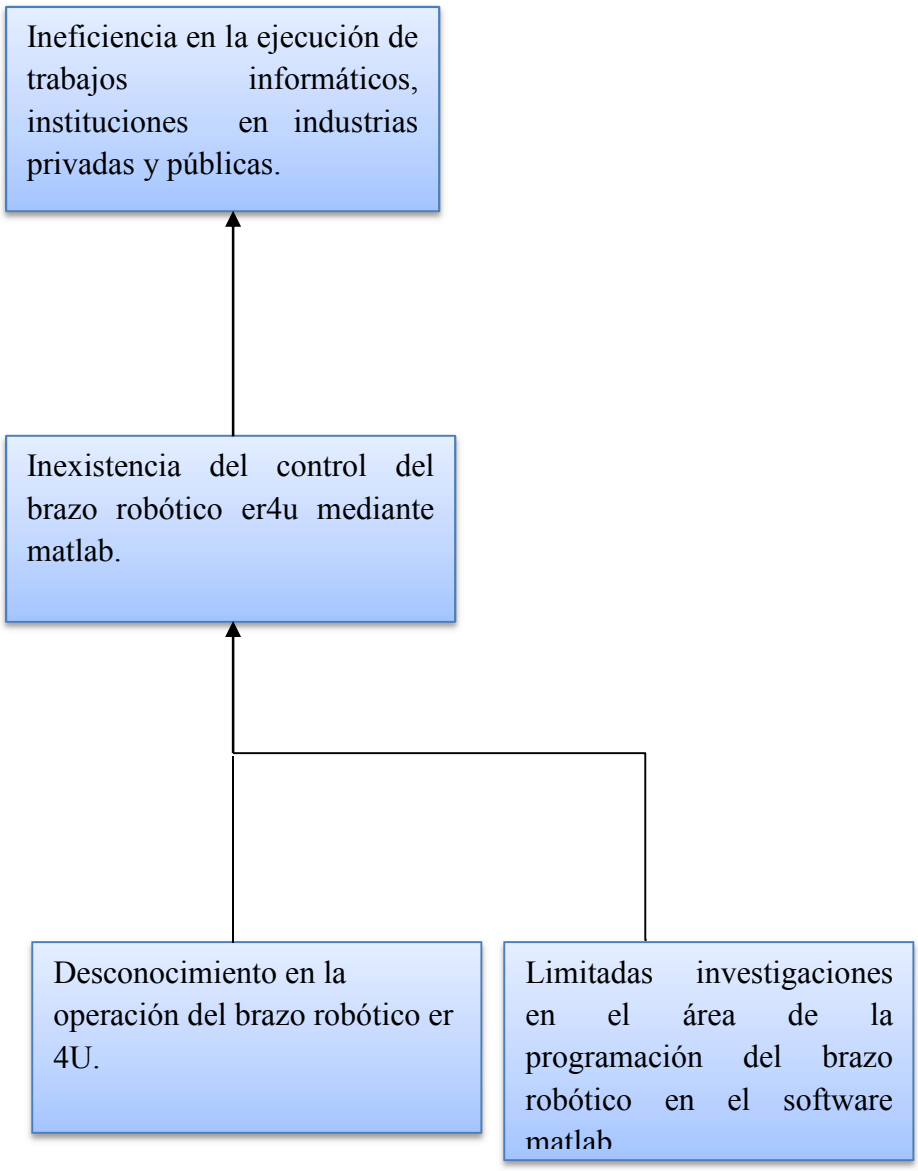
Grupos	Intereses	Problemas Percibidos	Recursos y Mandatos
Universitarios de la facultad de ciencias y tecnología de la carrera de Ing. informática.	Poner en práctica conocimientos percibidos en el brazo robot.	Desconocimiento en la operación del brazo robótico er 4U.	R: Alguna disponibilidad de usar la aplicación y el brazo robot si es confiable.
Docentes de la carrera de ciencias y tecnología de la carrera de Ing. informática.	Contar con herramientas prácticas en el área de robótica, para la manipulación de los mismos en el laboratorio.	Inexistencia del control del brazo robótico er4u mediante matlab. Limitadas investigaciones en el área de la programación del brazo robótico en el software matlab.	M: Proveer guía clara y precisa para la puesta en práctica del brazo robot.

UA Juan Misael Saracho	Fomentar la investigación en docentes y estudiantes. Para formar profesionales, capaces de contribuir al progreso de la sociedad.	Ineficiencia en la ejecución de trabajos informáticos, instituciones en industrias privadas y públicas.	M: Formar profesionales competentes e integrales que asimilen y transfieran, avances científico tecnológico de acuerdo a las exigencias del entorno.

Fuente: Elaboración Propia.

Tabla 1: Cuadro de Involucrados.

I.3.5. Árbol de Problemas



Fuente: Elaboración Propia.

Figura 1: Árbol de Problema.

I.3.6. Árbol de Objetivos

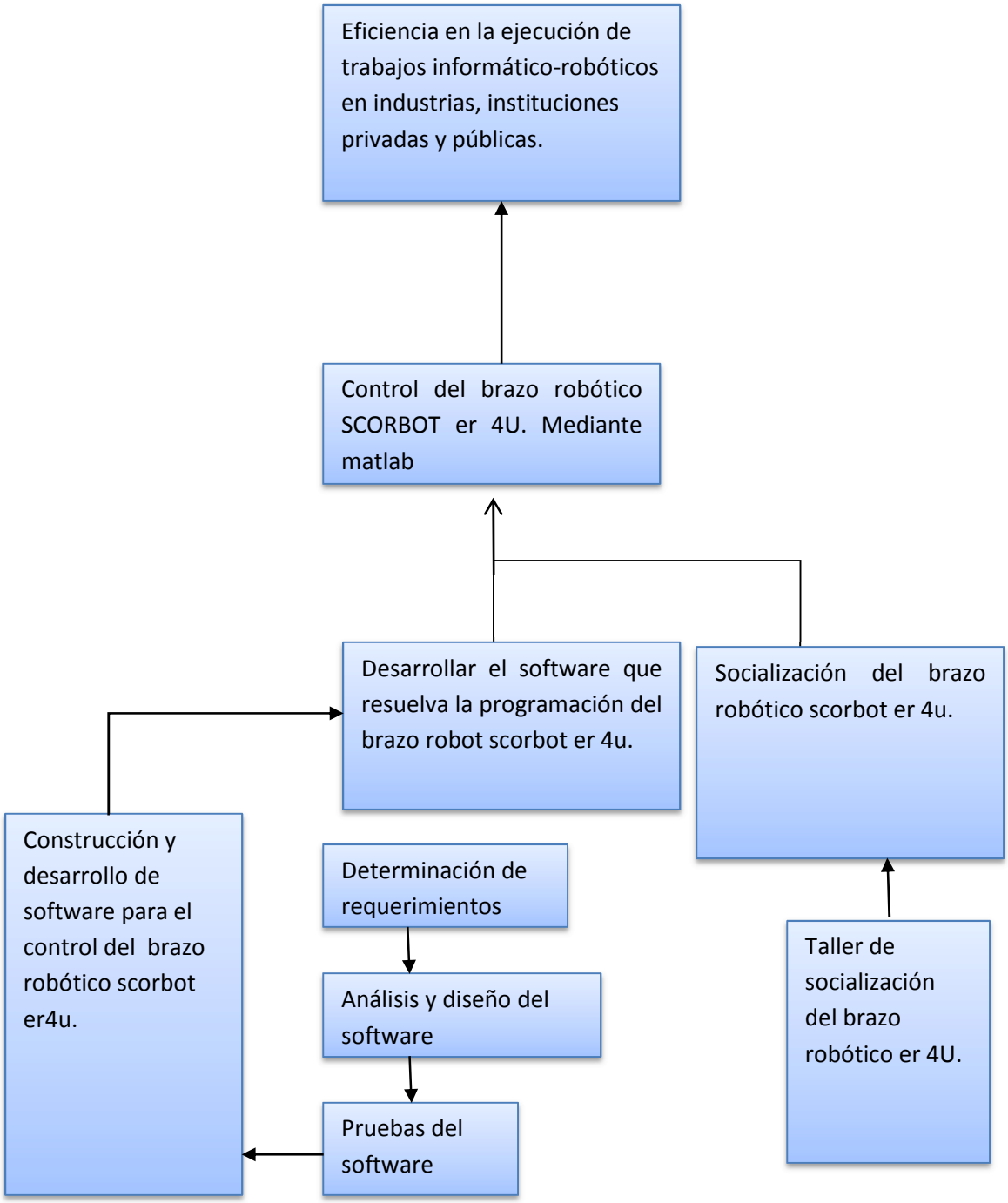


Figura 2: Árbol de Objetivos.

I.4. Situación planteada con y sin proyecto

Situación sin proyecto	Situación con proyecto
Estudios meramente teóricos de situaciones simuladas, en su máxima expresión, que serían hechas por ordenador. Basados en datos empíricos y supuestos que tienen un grado de incertidumbre.	Planteamiento de situaciones que se puede llevar a realizar basándonos en precedentes dejados en este proyecto, con datos matemáticos, herramientas utilizadas, y medios de verificación pasando desde la simulación de situaciones planteadas y llevarlas a la práctica con el uso del brazo robótico er 4U.

Fuente: Elaboración Propia.

Tabla 2: Situación con y sin proyecto

I.5. Objetivos

I.5.1. Objetivo General

Control del Brazo Robotico Scorbot er4u Mediante Matlab.

I.5.2. Objetivos Específicos

- Desarrollar el software que resuelva la programación del brazo robot scorbot er 4u.
- Investigación de la compatibilidad entre el brazo robot con matlab con posterior socialización del brazo robótico scorbot er 4u.

I.6. Marco Lógico del Árbol de Objetivos

Resumen Narrativo del Proyecto	**Indicadores**	**Medios de Verificación**	**Supuestos**
Fin: Eficiencia en la ejecución de trabajos informático-robóticos en industrias, instituciones privadas y públicas.	Para el años 2015 se prevé la presentación de un 4.16% trabajos investigativos relacionados en el área de robótica por alumnos interesados que mostraron aceptación por el trabajo. $\%=\frac{(Nº\ Intenciones\ de\ trabajo\ roboticos * 100)}{Nº \text{ de univ en taller III hasta el } 2015}$	Informe del encargado del laboratorio de robótica del número de proyectos desarrollados.	Los estudiantes de ingeniería informática desarrollan proyectos en base al scorbot er4u.
Objetivo General (Propósito) Control del brazo robótico SCORBOT er 4U. Mediante matlab implementado.	Al finalizar el proyecto, al menos un 50% del estudiantado de la carrera de ing. informática de los cursos de 4to y 5to año asistente a los cursos de socialización, expresan su aceptación con la demostración del producto a partir de agosto del 2013.	Informe del trabajo de investigación realizado por la docente de la materia de Taller III.	Participación y aceptación de estudiantes en los cursos de capacitación en un 85% y con ánimos de poner en práctica lo visto en la socialización.

	$\%=\frac{(Nº\ asistentes*100)}{Nº\ de\ alumnos\ inscritos}$		
Objetivos Específicos (Componentes). 1. Desarrollar el software que resuelva la programación del brazo robot scorbot er4u. **2.** Socialización del brazo robótico Scorbot er 4u	1. Al finalizar el proyecto en agosto de 2013 se pretende presentar un sistema funcional a más del 50% de los universitarios de 4to y 5to año de la carrera de ing. informática del curso taller, demostrando el sistema desarrollado. $\%=\frac{(Nº\ asistentes*100)}{Nº\ de\ alumnos\ inscritos}$ 2. Al finalizar el proyecto en diciembre de 2013 se pretende alcanzar socializar al menos un 50% del estudiantado universitario que cursa el 4to y 5to año de la carrera de informática. $\%=\frac{(Nº\ asistentes*100)}{Nº\ de\ alumnos\ inscritos}$	1 Documentación aprobada por los docentes de la materia de taller III. 2 Certificados de participación del curso de socialización del brazo robot scorbot er4u. 3 Listas de asistencia firmadas por los participantes. 4 Propaganda impresa de la socialización que	1. Adquisición del brazo robótico er4U como objeto de estudio. 2. Participaciones de estudiantes universitarios en el taller de socialización del proyecto.

		se llevara a cabo en los laboratorios de robótica. 5 Encuestas realizadas del curso taller por participantes del mismo. 6 Fotografías tomadas del curso taller.	
Actividades **1.** Taller de socialización sobre el brazo robótico scorbot er4u. **2.** Construcción y desarrollo de	**Ver punto I.10 (Presupuesto / Justificación)** **COSTO TOTAL Bs 158.040,40**	Plantilla de ejecución presupuestada.	1. Los desembolsos se realizan de acuerdo al cronograma tanto para seguir en el diseño del software como la manipulación del brazo robot.

software para el control del brazo robótico scorbot er4u.			2. Cumplimiento en la calendarización del proyecto la culminación de la investigación y socializar el mismo en las fechas previstas.

Fuente: Elaboración Propia.

Tabla 3 Matriz del Marco Lógico del Árbol de Objetivos.

I.7. Metodología de Trabajo

Para el inicio del proyecto, se programarán reuniones con el encargado del laboratorio de robótica, con el fin de poder tener acceso al brazo robótico scorbot er4u.

Se pretende dar uso al brazo robótico, probando que es posible la comunicación entre matlab y el brazo, desarrollando software para el control de este.

Para el desarrollo del sistema se utilizará la metodología RUP, el ciclo de vida clásico.

Para la etapa de programación del sistema, se utilizará el software matemático matlab, para el desarrollo de la aplicación que controle el brazo robótico más el toolbox[5].

Para la documentación del sistema se adoptará lo sugerido por la norma IEEE 830.

Se hará uso de las fases de cada metodología de acuerdo a los requerimientos del proyecto.

I.7.1. Ciclo de vida clásica

- Fase de Análisis.
- Fase de Diseño.
- Fase de Pruebas.
- Fase de Implementación.

I.7.2. Metodología UML

I. Análisis

a. Diagramas de casos de uso.

[5] mathworks. (s.f.). Obtenido de http://www.mathworks.com/matlabcentral/fileexchange/32699-matlab-toolbox-for-the-intelitek-scorbot/all_files (2013)

b. Diagramas de secuencia.
c. Diagramas de actividad.
d. Diagramas de componente.
e. Diagramas de navegación.
f. Diagrama de tiempo.
g. Diagrama de despliegue.

II. Diseño, Desarrollo, Producción, Instrumentación y Evaluación.

I.7.3. Curso de Socialización del "Control del Brazo Robótico Scorbot er4u"

a) Definición de las estrategias para el curso, se proponen estos temas.
   1. Introducción a la robótica industrial y manejo del brazo scorbot er 4u.

b) Elaboración del Cronograma de Actividades para el curso.
   1. La socialización tendrá una duración de una hora.
   2. Se avanzará con videos, diapositivas en PowerPoint y demostración con el brazo robot.

c) Desarrollo del curso.
   1. **Tema 1:** Introducción a la robótica industrial y manejo del brazo scorbot er 4u.
      - Descripción de la robótica industrial.
      - Diseño y control de una cedulá robótica.
      - Seguridad en instalaciones robotizadas.
      - Operaciones realizadas con el software desarrollado y el brazo robot scorbot er 4u.

I.7.4. Descripción y Relación de las Estrategias con los Objetivos

Estrategias	**Objetivos Específicos**
Trabajar con el brazo robótico	Compilación de temas matemáticos

corroborando que toda la parte matemática que se compila para su uso tenga el resultado deseado.	relacionados con cinemática y dinámica en robótica industrial.
Aplicar operaciones y procedimientos estandarizados organizar los datos, de forma que resulte información unificada.	Llevar un control de las operaciones del equipo, para la obtención de datos reales de rendimientos, que permitan optimizar recursos.
Facilitar el área correspondiente en donde se encuentra el brazo robótico.	Tener un lugar disponible para la realización de prácticas.

Fuente: Elaboración Propia.

Tabla 4: Descripción y Relación de las Estrategias con los Objetivos.

I.7.5. Cronograma de Actividades

	Nº	Actividad	Nº días	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
COMPONENTE I	**1**	▪ Recopilación de la información e investigar características técnicas del brazo robótico, más los accesorios utilizados y modelado de los casos de uso.	**30**			**x**									
	2	▪ Construcción de todos los diagramas UML del sistema. ▪ Diseño y elaboración de interfaces. ▪ Programación del sistema	**114**				**x**	**x**	**x**	**x**	**x**				
COMP	**3**	▪ Elaboración del material de apoyo para el operador.	**7**								**X**				
	4	▪ Elaboración de material y guías del curso de	**13**								**X**	**X**			

O N E T E II		socialización.													
		▪ Realizar capacitación.	**1**									**X**			
		▪ Realizar entrega de certificados de capacitación.	**1**									**X**			

Fuente: Elaboración Propia.

Tabla 5: Cronograma de Actividades.

I.8. Resultados esperados

a) Funcionamiento del brazo robótico.- al finalizar el proyecto se entregará acompañado con su correspondiente documentación el brazo robótico funcionando.

b) Manual de Usuario.- al finalizar el proyecto se elaborará un documento que sirva de guía para el usuario.

c) Socialización de las personas.- se llevará a cabo un taller de socialización del brazo robótico, haciendo conocer a la audiencia académica de la carrera de informática que irá dirigida preferentemente a los universitarios de 4to y 5to año el funcionamiento del brazo robótico mediante ecuaciones, teorías matemáticas.

I.9. Transferencia de resultados

a) Medios y estrategias para la transferencia de resultados

- Entrega del producto a la Universidad Autónoma Juan Misael Saracho.
- Socialización del producto.

b) Grupo de beneficiarios de los resultados.

- Universidad Autónoma Juan Misael Saracho.
- Población Universitaria de la Facultad de Ciencias y Tecnología de la carrera de Ingeniería Informática.

I.10. Presupuesto / Justificación

ITEM	RUBROS	Aporte Universidad	Otro Aporte	TOTAL (Bs.)
10000	**SERVICIOS PERSONALES**			
	12000 Empleados no Permanentes		22.376,00	22.376,00
	Sub total rubro			22.376,00
20000	**SERVICIOS NO PERSONALES**			

	21000. Servicios Básicos		1.800,00	1.800,00
	22000. Servicios de transporte			
	23000. Alquileres			
	24000. Mantenimiento y reparación		1.200,00	1.200,00
	25000. Servicios Profesionales y Comerciales		20.493,00	20.493,00
	Sub total rubro			23.493,00
30000	**MATERIALES Y SUMINISTROS**			
	31000. Alimentos y Productos Forestales			
	32000. Productos de Papel, Cartón e Impresos			
	33000. Textiles y Vestuario.			

	34000. Productos Químicos, Combustibles y Lubricantes			
	39000. Productos Varios.			
	Sub total rubro			
40000	**ACTIVOS REALES**			
	43000. Maquinaria y Equipo.		67.017,00	67.017,00
	46000. Descripción de estudios y proyectos para inversión			
	49000. Otros Activos			
	Sub total rubro			67.017,00
	TOTAL			112.886,00
	TOTAL + 40% Incentivo			158.040,40

Tabla 6: Presupuesto / Justificación

1) GRUPO 10000. SERVICIOS PERSONALES

a) SUB GRUPO 12000. Empleados no Permanentes

* Se refiere a gastos para remunerar a personas sujetas a contrato dependientes según la necesidad de cada entidad

Tabla 7: Empleados no Permanentes

Partida	Personal	Remuneración	Tiempo /meses	Total
12100	Jefe del proyecto	1.750,00	8	14.000,00
	Ingeniero de software	1.047,00	1	1.047,00
	Analista de sistemas	1.047,00	3	3.141,00
	Programador	1.047,00	4	4.188,00
Total				22.376,00

2) GRUPO 20000. SERVICIOS NO PERSONALES

b) SUB GRUPO 21000. Descripción de los gastos de servicios básicos

Partida	Tipo de servicio básico *	Costo	Tiempo mes	Costo Total
21100	Comunicación			
21200	Energía Eléctrica	276,00	6.5	1.800,00
21300	Agua			
21400	Servicios Telefónicos			
Total				1.800,00

Tabla 8: Descripción de los gastos de servicios básicos.

* Se refiere principalmente a los gastos por servicios; como: servicio de correo, radiogramas, servicio telefónico, fax, Internet.

c) SUB GRUPO 22000. Descripción de los gastos de viajes y transporte de personal

Partida	Personal	Lugar	Nº de viajes	Costo unitario*	Costo total
22100	Pasajes				
Total					

Tabla 9: Descripción de los gastos de viaje y transporte de personal

* En el caso de pasajes debe indicarse el costo de ida y vuelta (costo unitario), indicando el número de viajes.

Partida	Personal	Lugar	Duración (días)	Costo unitario*	Costo total
22200	Viáticos				
22300	Fletes y Almacenamientos				

22600	Transporte de Personal				
Total					
Total sub grupo 22000					

Tabla 10: Viáticos tomando en cuenta la escala establecida por la UAJMS

* En el caso de los viáticos, debe considerarse la escala establecida por la UAJMS.

d) SUB GRUPO 23000. Descripción de los gastos por concepto de alquileres de equipos y maquinarias

Partida	Alquiler de equipo y maquinaria	Costo unitario	Tiempo mes	Costo total
23100	Alquiler de Edificios			
23200	Alquiler de Equipos y Maquinaria			
23300	Alquiler de Tierras y Terrenos			
Total				

Tabla 11: Descripción de los gastos por concepto de alquiler de equipos y maquinaria.

* Se refiere principalmente a los gastos por el uso de edificios y equipos y maquinaria en general

e) SUB GRUPO 24000. Descripción mantenimiento y reparación

Partida	Mantenimiento y reparación de equipo y maquinaria	Costo unitario	Tiempo mes	Costo total
24100	Mantenimiento y Reparación de Edificios y Equipos	184	6.5	1.200,00
24300	Otros Gastos por Mantenimiento y Reparación			
Total				1.200,00

Tabla 12: Mantenimiento y reparación.

* Se refiere principalmente a los gastos por el mantenimiento y reparación de edificios y equipos y maquinaria en general

f) SUB GRUPO 25000. Descripción de los gastos en servicios profesionales y comerciales

Partida	Tipo de servicio profesional y comercial *	Cantidad	Costo unitario	Tiempo mes	Costo Total
25200	Estudios e Investigaciones	1	20.493,00	6.5	20.493,00

25500	Publicidad				
25600	Imprenta				
25700	Capacitación de Personal				
25800	Estudios e Investigaciones Para Proyectos de Inversión				
Total					20.493,00

Tabla 13: Gastos en servicios profesionales y comerciales.

* Se refiere a gastos por servicios profesionales de asesoramiento especializado, se incluyen, estudios, investigaciones, publicidad, imprenta, fotocopias, capacitación de personal y otros ejecutados por terceros.

3) GRUPO 30000. MATERIALES Y SUMINISTROS

g) SUB GRUPO 31000. Descripción de los gastos Alimentos y Productos Agroforestales

Partida	Tipo de material *	Cantidad	Costo/Unit	Total
31110	Refrigerios y Gastos Administrativos			
31200	Alimento para Animales			
31300	Productos Agroforestales y Pecuarios			

Total			

Tabla 14: Gastos en alimentos y productos agropecuarios.

* Se refiere a la adquisición de materiales y bienes como: alimentos y productos agroforestales, alimentos y bebidas para personas (indicar el total de refrigerios), alimentos para animales, productos pecuarios.

h) SUB GRUPO 32000. Descripción del gasto de Productos de Papel, Cartón e Impresos

Partida	Tipo de material *	Cantidad	Costo/Unit	Total
32100	Papel de Escritorio			
32200	Productos de Artes Gráficas, Papel y Cartón			
32300	Libros y Revistas			
32400	Textos de Enseñanza			
32500	Periódicos			
Total				

Tabla 15: Gastos de productos de papel.

* Se refiere a la adquisición de; papel y cartón en sus diversas formas y clases, impresos y publicaciones, periódicos, revistas, libros, fotocopias, etc.

i) SUB GRUPO 33000. Descripción del gasto en textiles y vestuario

Partida	Productos textiles y vestuarios	Cantidad	Costo/Unit	Total
33100	Hilados y Telas			
33200	Confecciones Textiles			
33300	Prendas de vestir			
33400	Calzados			
Total				

Tabla 16: Gastos en textiles y vestuario.

* Se refiere principalmente a los gastos por vestuario uniformes, ropa de trabajo

j) SUB GRUPO 34000. Combustibles, Productos Químicos, Farmacéuticos y Otros

Partida	**Combustibles, Productos Químicos, Farmacéuticos y Otros**	Cantidad	Costo/Unit	Total
34110	Combustibles y Lubricantes para Consumo			

34200	Productos químicos y Farmacéuticos			
34400	Productos de Cuero y Caucho			
34500	Productos de Minerales no Metálicos y Plásticos			
34600	Productos Metálicos			
34700	Minerales			
34800	Herramientas Menores			
Total				

Tabla 17: Combustibles, productos químicos, farmacéuticos y otros.

* Se refiere a gastos de combustibles, químicos, productos farmacéuticos, llantas etc.

k) SUB GRUPO 39000. Descripción del gasto en productos varios

Partida	Productos de cuero y caucho	Cantidad	Costo/Unit	Total
39100	Material de Limpieza			
39400	Instrumental Menor Médico - Quirúrgico			
39500	Útiles de Escritorio y de Oficina			
39700	Útiles y Materiales Eléctricos			
39800	Otros Repuestos y Accesorios			
Total				

Tabla 18: Descripción del gasto en productos varios.

*Se refiere principalmente a los gastos por productos de limpieza, todo le referente a la funcionamiento de la oficina en material de escritorio.

4) GRUPO 40000. ACTIVOS REALES

l) SUB GRUPO 43000. Descripción del gasto de Maquinaria y Equipo

Partida	Tipos de productos	Cantidad	Costo/Unitario	Total
43100	Equipo de Oficina y Muebles			
43200	Maquinaria y Equipo de Producción	1	67.017,00	67.017,00
43300	Equipos de Transporte, Tracción y Elevación			
43400	Equipo Médico y de Laboratorio			
43600	Equipo Educacional y Recreativo			
43700	Otra Maquinaria y Equipo			
Total				67.017,00

Tabla 19: Gasto de maquinaria y equipo

* Se refiere principalmente a los gastos por muebles y enseres, equipo de oficina, comunicación, equipamiento.

m) SUB GRUPO 46000. Descripción de estudios y proyectos para inversión

Partida	Productos textiles y vestuarios	Cantidad	Costo/Unit	Total
46100	Para Construcción de Bienes de Dominio Privado			
Total				

Tabla 20: Estudios y proyectos para inversión.

* Se refiere principalmente a los gastos por servicios de terceros para la realización de investigaciones y otras actividades técnico – Profesionales necesarias para la construcción y mejoramiento de bienes.

n) SUB GRUPO 49000. Descripción del gasto de Otros Activos

Partida	Tipos de productos *	Cantidad	Costo/Unit	Total
49100	Activos Intangibles			
49200	Compra de Bienes Muebles Existentes (Usados)			
49300	Semovientes y otros Animales			
49900	Otros Activos			
Total				

Tabla 21: Descripción del gasto de otros activos.

* Se refiere a los gastos en la compra de sotfwear, licencias.

4. Curriculum Vitae (del Director y Equipo de Trabajo: llenar una ficha para cada uno de ellos)

4.1 Antecedentes personales

SUBELZA Apellido Paterno	**FLORES** Apellido Materno	**DANIEL** Nombre	**5817951** C.I**.**
09/02/1986 Fecha de nacimiento	M...SI.. F..... **Sexo**	B. Luis Pizarro **Dirección**	
Tarija Ciudad	466-46852 **Teléfono Domicilio**	72976542 **Celular**	daniels_0076@hotmail.com **Correo electrónico**

Tabla 22: Antecedentes personales.

4.2 Antecedentes académicos

Título obtenido	Universidad	País	Año

Carrera Ing. Informática	Universidad Autónoma Juan Misael Saracho.	Bolivia	2013

Tabla 23: Antecedentes académicos.

4.3 Participación en proyectos de investigación

Título proyecto	Institución	Cargo	Año

Tabla 24: proyectos de investigación.

4.4 Publicaciones realizadas (libros, revistas, compendios y otros)

Autor	Tipo de publicación, Año, título, volumen, páginas, editorial

Tabla 25: Publicaciones realizadas.

* Indique las publicaciones realizadas durante los últimos 5 años

4.5 Presentaciones realizadas

Título	Nombre del evento, lugar, fecha, año

Tabla 26: Presentaciones realizadas.

* Incluir hasta 5 presentaciones de los últimos 5 años

4.6 Antecedentes en docencia

a) Experiencia Docente

Experiencia Docente		
Institución	Dedicación	Años

Tabla 27: Experiencia Docente.

b) Jerarquía Docente (en la presente gestión, y de gestiones anteriores)

Categoría docente	Carrera/Departamento	Tiempo dedicación	Gestión

Tabla 28: Jerarquía Docente

II. Componentes

II.1.Componente 1: Control del Brazo Robótico Scorbot er4u Mediante Matlab.

II.1.1. Información general del brazo robot scorbot er4u

El brazo robótico SCORBOT er4u es una herramienta educativa, es un motor articulado vertical, similar a un brazo, con 5 articulaciones (ejes) para su movimiento y pinzas.

El software actual que maneja el brazo robot es de código cerrado, con un entorno de control definido, sin posibilidades de modificación para la operación del brazo robot.

Se pretende realizar la investigación sobre la comunicación entre el brazo robot y el programa matemático matlab mediante una aplicación de código abierto y un toolbox que operara en el área específica de control del brazo robot mediante comandos de cosola, explorando el toolbox se quiere marcar las aplicaciones y limitaciones a las que se podría llegar. Pretendiendo llegar así un control más profundo al que actualmente tiene el software ofrecido por la empresa Intelink desarrolladora del brazo robot scorbot er 4u y que estará abierta a modificación por parte de los operadores para "personalizar" el entorno de operación siempre y cuando se tenga conocimiento en programación en C++ y teniendo presente la sintaxis manejado en matlab.

II.1.1.1. Definición del problema

El proyecto de investigación "Control del brazo robótico Scorbot er4u mediante matlab" tiene la finalidad de hacer uso de herramientas que están en desuso, como ser el brazo robotico er4u para que la población universitaria de la facultad de ciencias y tecnología en la carrera de ingeniería informática pueda realizar otros aportes, avances en el campo de la robótica.

II.1.1.2. Características del brazo robot Scotbot er4u

Los movimientos del brazo robot son posibles bracias a un motor de cd[1]. Se describe el movimiento realizado por eje.

Nro del eje	Nombre articulación	Movimiento
1	Base	Rota el cuerpo
2	Hombro	Levanta y baja el ante brazo
3	Codo	Levanta y baja el brazo
4	Elevación	Levanta y baja la pinza
5	Giro	Gira pinza
6	Pinza	Abre y cierra la pinza

Fuente: Elaboración propia.

Tabla 1: Ejes y movimientos.

II.1.1.3. Características del Controlador-USB

El controlador-USB sirve para operar el brazo robot y los accesorios. Se lo conecta a un computador vía conector USB. Posee una fuente de alimentación que suministra 24 Vcd. Además posee puertos de entrada y salida, digitales y análoga, para conectarse a accesorios adicionales como sensores, interruptores, actuadores, transmisores.

II.1.2. Lenguaje de programación matlab-VRML

[1] Corriente Directa.

Desde las primeras versiones MATLAB incorpora una característica muy interesante: la capacidad para programar. En efecto, es posible crear archivos que contengan las operaciones que se desean realizar. Además es posible incorporar nuevas funciones a MATLAB realizadas por el propio usuario.

La programación de este proyecto se lleva a cabo mediante un lenguaje que es muy parecido a lenguaje de alto nivel como BASIC o C. esto permite que el usuario pueda agrupar sentencias que se utiliza frecuentemente dentro de un programa que pueda ser involucrado posteriormente.

Para el diseño y construcción de la pantalla de simulación se utilizará VRML (Virtual Reality Modelling Languaje). La herramienta de realidad virtual (VRML) es una solución para las vistas e interacción con sistemas dinámicos de entornos de realidad virtual tres dimensiones.

Y por último la librería MTIS-Toolbox que se investiga si puede producir la comunicación entre la pc y el control-usb para comunicarse con el brazo robot.

Es una librería educacional de código abierto.

Código fuente de la librería MTIS-Toolbox

Función BSEPR=ScorGetJt

```
function BSEPR=ScorGetJt

% FUNNCION BSEPR=ScorGetJt

% devuelve la Base, Hombro, Codo Giro, Vuelta

% asignando el Angulo en radianes

%

F = 0; B = 0.0; S = 0.0; E = 0.0; P = 0.0; R = 0.0;
```

```
try
   [F,B,S,E,P,R]=calllib('RobotDll','RGetBSEPR',B,S,E,P,R);
catch
   disp('ScorGetBSEPR Failed: Teach Pendant much be in Auto Mode')
end
BSEPR=pi/180*[B/1000 -S/1000 -E/1000 -P/1000 R/1000];
```

Función RVal=ScorCreateVector(Name,Len)

```
function RVal=ScorCreateVector(Name,Len)
% RVal=ScorCreateVector(Name,Len)
% Crea un vector para el Scorbot.
% en el que se puede almacenar posiciones.
% Name (Entre comillas) es una cadena (String)
% para el vector de 16 o menos caracteres.
% Len es el numero puntos posibles en el vector < 10000
% devuelve 1 si tiene éxito
% utilizado durante la inicialización.
[R,NVal]=calllib('RobotDll','RDefineVector',Name,Len);
RVal=R;
```

Función ScorBlockUntilMotionComplete(XYZPR)

```
function  ScorBlockUntilMotionComplete(XYZPR)
% Esta es una función que se utiliza para bloquear comandos
```

```
% hasta que termine el comando actual que se ese ejecutando.
% Solo funciona en modo auto. Llame a esta función después de
% que por vía "confirmación" ese comando no ha fallado.
DistTol = 1.0;  %  10 * Linear resolution in cm
AngleTol = 5.0;  % 10 * angular resolution in degrees
PauseTime = 0.01;  % in seconds, is this really needed?
CurrentXYZPR = ScorgetXYZPR;
HowFarToGo = norm( CurrentXYZPR(1:3)-XYZPR(1:3) );
PitchToGo =  abs( CurrentXYZPR(4)-XYZPR(4) );
RollToGo =  abs( CurrentXYZPR(5)-XYZPR(5) );

while((calllib('RobotDll','RIsMotionDone'))==0)        ||    HowFarToGo>
DistTol  ||  PitchToGo>AngleTol || RollToGo>AngleTol
   CurrentXYZPR = ScorgetXYZPR;
   HowFarToGo = norm( CurrentXYZPR(1:3)-XYZPR(1:3) );
   PitchToGo =  abs( CurrentXYZPR(4)-XYZPR(4) );
    RollToGo =  abs( CurrentXYZPR(5)-XYZPR(5) );
    pause(PauseTime);
end
```

Funcion ScorCartMove(XYZPR)

```
% FUNCION confirmation = ScorCartMove(XYZPR)
% ScorCartMove(XYZPR) se mueve a una posicion de destino definida
en coordenadas    % cartesianas.
```

```
% definidas por el vector XYZPR
% XYZPR - 1x5 vector de x, y, z en cm; giro, vuelta en grados.
% confirmation = 0 es que fallo el movimiento.  XYZPR fuera del área del trabajo es un error % común.
function confirmation = ScorCartMove(XYZPR)
% cuan cerca tienes que esta de la posición de un objetivo antes de aceptar otro.
ScorRequestPendantMode('auto');
confirmation = ScorAddToVec(1000, XYZPR);
if confirmation
    confirmation = ScorMoveToPt(1000, 'L');
end
%% Bloque del comando hasta que la ejecución esté completa.
if(confirmation==1)
   ScorBlockUntilMotionComplete(XYZPR);
end
end
```

Función ScorDeltaJtMove(deltaBSEPR)

```
% confirmation = ScorDeltaJtMove(deltaBSEPR)
% incremento (delta) de cambios en ángulos de articulaciones.
% deltaBSEPR - 1x5 vector de cinco (Base, Hombro, Codo, Giro, Vuelta)
function confirmation = ScorDeltaJtMove(deltaBSEPR)
```

```
deltaBSEPR(2) =-deltaBSEPR(2);

deltaBSEPR(3) =-deltaBSEPR(3);

deltaBSEPR(4) =-deltaBSEPR(4);

ScorRequestPendantMode('auto')

xc=ScorGetXYZPR;                    %posición actual del vector
[x,y,z,giro,vuelta]

thc=ScorX2Deg(xc);          %calcula vector angular articular actual.

thf=thc+deltaBSEPR;                 %posición de movimiento del vector
final con Angulo                                                    %
%articular.

xf=ScorDeg2X(thf);                  %posición final del vector de
movimiento[x,y,z,giro,vuelta]

confirmation = ScorAddToVec(1000, xf);

if confirmation

   confirmation = ScorMoveToPt(1000, 'J');

end



if(confirmation==1)

  ScorBlockUntilMotionComplete(xf);

end
```

Funcion ScorDeltaJtMoveWristLevel(deltaBSE)

```
% confirmation = ScorDeltaJtMoveWristLevel(deltaBSE)

% incremental (delta) cambio en los primeros tres articulaciones angulares

% deltaBSEPR - 1x5 vector de cinco (Base, Hombro, codo, Giro, Vuelta)

% conjuntos de incrementos en radianes

% Posición del robot para define la posicion final deseada movimientoangular de las % articulaciones.

function confirmation = ScorDeltaJtMoveWristLevel(deltaBSE)

deltaBSE = 180/pi* deltaBSE; %degs

deltaBSE(2) =-deltaBSE(2);

deltaBSE(3) =-deltaBSE(3);

deltaBSE(4) = -(deltaBSE(2)+  deltaBSE(3) );

deltaBSE(5) = 0;

ScorRequestPendantMode('auto')

calllib('RobotDll','RIsMotionDone')

xc=ScorGetXYZPR;                    %posición actual del vector [x,y,z,giro,vuelta]

thc=ScorX2Deg(xc);          %calcula vector angular articular actual.

thf=thc+deltaBSE';              %posición final del vector angular

xf=ScorDeg2X(thf);                  % posición final del vector de movimiento[x,y,z,giro,vuelta]

%manteniendo la muñeca

confirmation = ScorAddToVec(1000, xf);
```

```
if confirmation
    confirmation = ScorMoveToPt(1000, 'J');
end
if(confirmation==1)
   ScorBlockUntilMotionComplete(xf);
end
```

Función ScorDigitalOutput(D,OnOff)

```
function ScorDigitalOutput(D,OnOff)
%Emite un valor de salida digital
%el valor es el bit que se active / desactiva
%si OnOff=0 el bit se apaga
%si OnOff=1 el bit se prende
if(OnOff==0) calllib('RobotDll','RDigOff',D);
else if(OnOff==1) calllib('RobotDll','RDigOn',D);
    end
end
```

Función ScorHome

```
function ScorHome
```

```
% FUNCION ScorHome
% no hay entradas ni salidas
% rutina de 'Home' pide a todos los ejes que vuelvan a su posición de inicio.
ScorRequestPendantMode('auto');
pause(0.2);
disp('Homing! Please wait...')
HomeFail = calllib('RobotDll','RHome',65);
pause(1);
Home = [16.9030         0   50.4328  -63.5480         0];
XYZPR = ScorGetXYZPR();
if norm(Home(1:3) - XYZPR(1:3))>5
   disp('Homing Failed.   Try Again?');
else
   disp('Finished homing.')
end
ScorControlEnable(1);
```

Función ScorInit

```
function ScorInit
% FUNCION SCORINIT
% No tiene entradas ni salidas
% carga el archivo especial RobotDll e inicializa el Robot.
% la luz cambiará a verde en la caja de control
```

```
% El control se pone "on' para todos los ejes.
% se recomienda que esté en su Home el robot.
warning off
disp('Loading Dll Files');
if  ~libisloaded('RobotDll')|| ~libisloaded('RobotDll.h')
   try
     loadlibrary('N:\ES450\Code\RobotDll','N:\ES450\Code\RobotDll.h');
   catch
      try

loadlibrary('M:\ES450\Code\RobotDll','M:\ES450\Code\RobotDll.h');
      catch
         try
         loadlibrary('RobotDll','RobotDll.h');
         catch
            beep
            disp('ERROR: DID NOT FIND ROBOTDLL !!!')
         end
      end
   end
end
disp('Initializing USB interface ...');
calllib('RobotDll','RInitialize');
pause(2);
```

```
while (calllib('RobotDll','RIsInitDone')==0)
    % Bloquea antes de inicializar
   pause(.1);
end;
disp('POWER Light turned Green if successful.')
fprintf('\n')
calllib('RobotDll','RDefineVector','USNA',1000);
pause(1)
disp('Ready!')
fprintf('\n')
disp('Turn teach pendant to AUTO')
disp('and home the robot with ScorHome.')
warning on
```

Función ScorJtMove(BSEPR)

```
% FUNCION confirmation =ScorJtMove(BSEPR)
% ScorJtMove(BSEPR) se mueve a una posicion definida por
% el vector 1x5 de cinco datos
% [Base, Hombro, Codo, Giro, Vuelta]
function confirmation = ScorJtMove(BSEPR)
BSEPR(2:4) = -1*BSEPR(2:4);
ScorRequestPendantMode('auto');
xf=ScorDeg2X(BSEPR)          %vector con la posición final
[x,y,z,pitch,roll]
```

```
confirmation = ScorAddToVec(1000, xf)
if confirmation
    confirmation = ScorMoveToPt(1000, 'J')
end
if(confirmation==1)
   ScorBlockUntilMotionComplete(xf);
end
```

Función ScorJtMoveWristLevel(BSE)

```
% FUNCION confirmation =ScorJtMoveWristLevel(BSE)
% Se mueve a una posición definida por el
% vector 1x3 de ángulos articulares en radianes
% [Base, Hombro, Codo]
Mantiene el Giro y Vuelta = 0 (nivel de la muñeca)
function confirmation = ScorJtMoveWristLevel(BSE)
BSE = 180/pi* BSE; %degs
% manteniendo el nivel de la muñeca
BSE(end+1) = -( BSE(2) + BSE(3) );
BSE(end+1) = 0;
confirmation =ScorJtMove(BSE);
```

Función ScorRequestPendantMode(desiredMode)

```
function []= ScorRequestPendantMode(desiredMode)
% FUNCTION []= REQUESTPENDANTMODE(desiredMode);
% Una función de utilidad que comprueba el modo de la unidad de programación
% y espera (posiblemente para siempre) hasta que el usuario
% cambia al modo deseado.
if strcmp(desiredMode, 'teach')
   modeFlag = 1;
elseif strcmp(desiredMode, 'auto')
   modeFlag = 0;
else
   disp('Warning: RequestPendantMode defaulting to Auto as desired mode!')
   desiredMode='auto';
   modeFlag = 0;
end
Mode= calllib('RobotDll','RIsTeach');
while(Mode~=modeFlag)
   beep; beep;
   disp('Error: Please Switch Pendant Mode to:  ');
   fprintf(desiredMode);
   fprintf('\r\n');
```

```
    pause(1)
    Mode=calllib('RobotDll','RIsTeach');
    if Mode==modeFlag
        disp('Pendant now in proper mode! Proceeding...')
    end
end
```

Función ScorSafeShutDown

```
function ScorSafeShutDown
% ScorSafeShutDown Ejecute este comando antes de salir de Matlab
% Scorbot pierde poder tan pronto se salga de  matlab
% que se pondrá de nuevo en la posición neutral.
Home = [16.9030         0    50.4328  -63.5480         0];
ScorCartMove(Home);
ScorControlEnable(0);
```

Función ScorUnload

```
function ScorUnload
% Carga las librerías DLL.
unloadlibrary('RobotDll');
```

Función ScorWaitUntilDone()

```
function ScorWaitUntilDone()
```

```
% similitud con la función ScorBlockUntilMotionComplete
   BSEPRold = ScorGetBSEPR;
   GripOld = ScorGetGripper;
   busy=1;
   while busy ==1;
      Ready = calllib('RobotDll','RIsMotionDone');
      BSEPR = ScorGetBSEPR;
      EndMovement = norm(BSEPR-BSEPRold);
      Gripper = ScorGetGripper;
      GripperMovement = abs(Gripper - GripOld);
      fprintf('R: %d, E: %f, G: %f \n',[Ready, EndMovement,
GripperMovement])
      if Ready==1 && EndMovement<1.7453e-004 &&
GripperMovement<.1
         busy =0;
      else
         GripOld = Gripper;
         BSEPRold = BSEPR;
      end
   end
```

Función XYZPR=ScorGetCart

```
function XYZPR=ScorGetCart
```

```
% XYZPR=ScorGetCart
% devuelve X,Y,Z (cms) y giro y vuelta (degs)
% como un vector de 5 X 1;
% Si se presenta algún error, devuelve nan
F = 0; X = 0.0; Y = 0.0; Z = 0.0; P = 0.0; R = 0.0;
try
[F,X,Y,Z,P,R]=calllib('RobotDll','RGetXYZPR',X,Y,Z,P,R);
XYZPR(1)=X/10000;
XYZPR(2)=Y/10000;
XYZPR(3)=Z/10000;
XYZPR(4)=P/1000;
XYZPR(5)=R/1000;
catch
   XYZPR = nans(1,5);
end
```

Función [XYZPR, confirmation]=ScorCapturePose(Pt)

```
function [XYZPR, confirmation]=ScorCapturePose(Pt)
% FUNCTION [XYZPR, confirmation]=ScorCapturePose(Pt)
% Graba actual posición como un punto # Pt (1 a 999) en el controlador
% y retorna la posición en el entorno de trabajo MATLAB como
XYZPR, un vector 1x5.
% X,Y,Z en cms, y Pitch Roll en deg
```

```
% confirmación de un cero para un punto que esté fuera del perímetro
% área de trabajo o fuera del rango 1 a 999,
% confirmación como  1 si tiene éxito.
ScorRequestPendantMode('teach');
XYZPR=ScorGetXYZPR;
F=ScorAddToVec(Pt,XYZPR);
if F >0
   confirmation = 1;
else
   confirmation = 0;
end
disp('We suggest you switch the teach pendant back to AutoMode before executing any movements.')
```

Función [theta]=ScorInverseKinematics(x,y,z)

```
function [theta]=ScorInverseKinematics(x,y,z)
% Asume que la pinza este apuntando directamente hacia abajo (-pi/2).
% THETA=ScorInverseKinnematics(X,Y,Z) conversion de Cartesiano end-effector
% positions to joint angles.
%
% - X,Y, y Z son variables escalares en metros.
% - THETA es un vector 3x1 en radianes.
zw=z+0.145;
```

```
a=sqrt(x^2+y^2)-.016;
b=zw-0.349;
d=sqrt(a^2+b^2);
cosgam=(0.221^2+0.221^2-d^2)/(2*0.221*0.221);
singam=-sqrt(1-cosgam^2);
gamma=atan2(singam,cosgam);
theta3=pi-gamma;
beta=atan2(0.221*sin(theta3),0.221+0.221*cos(theta3));
alpha=atan2(b,a);
theta2=alpha-beta;
theta1=atan2(y,x);
theta=[theta1;theta2;theta3];
```

Función cm=ScorGetGripper

```
function cm=ScorGetGripper
% FUNCTION cm=ScorGetGripper
% devuelve posicion en CM
% 0 cm si está completamente cerrada,
% completamnte abierta alrededor de los 7cm
cm=calllib('RobotDll','RGetJaw');
cm=cm/10;
```

Función cnts=ScorDeg2Cnts(theta)

```
% FUNCTION cnts=ScorDeg2Cnts(theta)
% cnts=ScorDeg2Cnts(theta) convierte angulos relativos de  D-H
% a SCORBOT-ER 4u contadores codificados
% cts - 1x5 vector de contador del motor
% Theta - vector de 1x5 de ángulos de la articulación DH relativos en grados
% Ver también ScorCnts2Deg
function cnts=ScorDeg2Cnts(theta)
kb=-141.8888;
kse=-113.5111;
offs=120.27;
offe=-25.24;
kw=-27.9;
offw=63.57;
cnts(1)=kb*theta(1);
cnts(2)=kse*(offs+theta(2));
cnts(3)=kse*(offe-theta(2)-theta(3));
cnts(4)=kw*(offw-theta(2)-theta(3)-theta(4)-theta(5));
cnts(5)=kw*(theta(2)+theta(3)+theta(4)-theta(5)-offw);
cnts=cnts(:)';
```

Función confirmation=ScorAddToVec(Pt,XYZPR)

```
function confirmation=ScorAddToVec(Pt,XYZPR)
%  confirmation=ScorAddToVec(Pt,XYZPR)
% añade una nueva posicion absoluta XYZPR (cm / deg)
% Pt es el indice en el vector de poses almacenada en robot 1-999
% devuelve 1 si punto fue adicionado. Puntos no seran adicioados (devuelve 0)
% si el punto resulta fuera del rango del entorno de trabajo.
FVal=0;
for i=1:3
   V1(i)=fix(XYZPR(i)*10000);
end
for i=4:5
   V1(i)=fix(XYZPR(i)*1000) ;
end;
%disp('XYZ Absolute');
[FVal,NVal]=calllib('RobotDll','RAddToVecXYZPR','USNA',Pt,0,V1(1),V1(2),V1(3),V1(4),V1(5));
confirmation=(FVal>0);
```

Función confirmation=ScorControlEnable(OnOff)

```
function confirmation=ScorControlEnable(OnOff)
% confirmation=ScorControlEnable(OnOff)
```

```
% Puede ayudar a recuperar después de un accidente. También se puede hacer con Teach % Pendant
% CONTROL ON/OFF <ENTER>
%si OnOff = 0 todas las juntas estarán apagadas
%if OnOff = 1 todas las juntas estarán prendidas
%devielve 1 si tiene éxito.
confirmation=calllib('RobotDll','RControl',int8('A'),OnOff);
```

Función confirmation=ScorDeltaCartMove(deltaXYZPR)

```
function confirmation=ScorDeltaCartMove(deltaXYZPR)
% confirmation=ScorDeltaCartMove(dXYZPR)
% Comandos de un movimiento gradual (delta) en XYZPR
% formato.  Delta XYZPR es un 1 X 5 vector donde
% X,Y,Z está en cm y P,R en deg
% devuelve 1 si el punto fue exitoso. Falla (confirmation =0)
% si el punto resulta fuera del rango del entorno de trabajo
ScorRequestPendantMode('auto')
XYZPRold = ScorGetXYZPR;
confirmation = ScorAddToVec(1000, XYZPRold + deltaXYZPR);
if confirmation
   confirmation = ScorMoveToPt(1000, 'L');
end
if(confirmation==1)
```

```
   ScorBlockUntilMotionComplete(XYZPRold + deltaXYZPR);
end
```

Función confirmation=ScorMoveToPt(Pt,LorJ)

```
function confirmation=ScorMoveToPt(Pt,LorJ)
% confirmation=ScorMoveToPt(Pt,LorJ)
% cambia a un punto almacenado internamente en el scorbot
% Pt = 1 a 999
% LorJ es un string: Lineal 'L' o conjunto 'J'
% 'L' conecta los dos puntos con una línea recta en el espacio cartesiano
% 'J' conecta los puntos con una línea recta en el espacio de la articulación, produciendo
% trayectorias aparentemente curvas.
% Puntos solo se pueden almacenar en formato XYZPR
% una pequeña desviacion puede causar que falle (imposible de ejecutar)
ScorRequestPendantMode('auto')
if(LorJ == 'L')
   confirmation=calllib('RobotDll','RMoveLinear','USNA',Pt);
else
   if(LorJ == 'J')
      confirmation=calllib('RobotDll','RMoveJoint','USNA',Pt);
   end
end
 if(confirmation==1)
```

```
    while((calllib('RobotDll','RIsMotionDone'))==0) % only properly blocks new commands in Auto mode
    end
end
```

Función confirmation=ScorSetGripper(cm)

```
function confirmation=ScorSetGripper(cm)
% FUNCION confirmation=ScorSetGripper(cm)
% ajusta el ancho del agarre en cm (centimetros).
% abre la pinza totalmente si cm= -1, cierra totalmente si cm=0,
% Abierta total a 7 cm
ScorRequestPendantMode('auto');
if(cm==0)
   confirmation=calllib('RobotDll','RGripClose');
elseif(cm== -1)
   confirmation=calllib('RobotDll','RGripOpen');
else
   % convierte cm a milimetros
   confirmation=calllib('RobotDll','RGripMetric',round(10*cm));
end

if calllib('RobotDll','RIsMotionDone')==-1
   pause(1)
else
```

```
   while( calllib('RobotDll','RIsMotionDone')==0)
      % Línea de comando de bloqueo hasta que...no funcione el modo
      % de aprendizaje
      %disp('.')
   end
end
```

Función confirmation=ScorSetMovetime(t)

```
function confirmation=ScorSetMovetime(t)
% confirmation=ScorSetMovetime(t)
% Establecer tiempo de movimiento (segundos), establecer la velocidad de movimiento.
confirmation=calllib('RobotDll','RSetTime',1000*t);
```

Función confirmation=ScorSetSpeed(PrcSpd)

```
function confirmation=ScorSetSpeed(PrcSpd)
% FUNCTION confirmation=ScorSetSpeed(PrcSpd)
% coloca la velocidad en porcentaje de velocidad maxima, deben de ser enteros entre 0 and %100
% confirmation = 1 Exito, 0 Falla
confirmation=calllib('RobotDll','RSetSpeed',round(PrcSpd));
```

Función enc2theta(q)

```
% ENC2THETA
```

```
% THETA = ENC2THETA(Q) convierte Scorbot cuentas del codificador con
% respect ángulos artculares DH.
% - Q es un vector 5x1 enrecuentos del codificador..
% - THETA es un vector 5x1 en radianes.
function theta = enc2theta(q)
k=[141.8888, -113.5111, 113.5111, 55.8 -55.8];
theta(1)=q(1)/k(1);
theta(2)=q(2)/k(2) + 120.3;
theta(3)=q(3)/k(3) + 25.3 - theta(2);
theta(4)=(q(4)-q(5))/k(4)+26.5-theta(3)-theta(2);
theta(5)=(q(4)+q(5))/k(5);
theta=(pi/180)*theta(:);%convierte a radianes.
```

Función q=theta2enc(theta)

```
% THETA2ENC
% Q = THETA2ENC(THETA) convierte relativos DH ángulos de las articulaciones a
% Scorbot cuentas del codificador.
% - Q es un vector 5x1en cuentas del codificador.
% - THETA es un vector 5x1 en radianes
function q=theta2enc(theta)
thetad=(180/pi)*theta;%convierte a grados
```

```
k=[141.8888, -113.5111, 113.5111, 55.8 -55.8];
q(1)=k(1)*thetad(1);
q(2)=k(2)*(thetad(2)-120.3);
q(3)=k(3)*(thetad(2)+thetad(3)-25.3);
q(4)=(k(4)*(thetad(2)+thetad(3)+thetad(4)-26.5)+k(5)*thetad(5))/2;
q(5)=k(5)*thetad(5)-q(4);
q=round(q(:));%hace enteros
```

Función theta=ScorCnts2Deg(cts)

```
% FUNCTION BSEPR=ScorCnts2Deg(cts)
% BSEPR=ScorCnts2Deg(cts) convierte SCORBOT-ER 4u vector
% a un ángulo relativo D-H
% cts - vector 1x5
% BSEPR - vector 1x5 a un ángulo relativo D-H en grados.
% [Base Shoulder Elbow Pitch Roll]
function theta=ScorCnts2Deg(cts)
kb=-141.8888;
kse=-113.5111;
offs=120.27;
offe=-25.24;
kw=-27.9;
offw=63.57;
theta(1)=cnts(1)/kb;
```

```
theta(2)=cnts(2)/kse-offs;
theta(3)=offs+offe-(cnts(2)+cnts(3))/kse;
theta(4)=(cnts(5)-cnts(4))/(2*kw)+cnts(3)/kse+offw-offe;
theta(5)=(cnts(4)+cnts(5))/(-2*kw);
theta=theta(:)';
```

Función theta=ScorX2Deg(x)

```
% FUNCTION BSEPR=ScorX2Deg(XYZPR)
% theta=ScorX2Deg(x) calcula en ángulo del vector D-H, theta,
% como una función del vector end-effector, x
% (Nota: se asume un "codo hacia abajo"  configuración del brazo)
% x - 1x5 vector de x, y, z en cm; pitch, roll en grados
% theta - 1x5 vector de cinco ángulos articulares D-H
function theta=ScorX2Deg(x)
d1=34.9;
a1=1.6;
a2=22.1;
a3=22.1;
d5=14.5125;
pitch=pi*x(4)/180;%en radianes
theta(1)=(180/pi)*atan2(x(2),x(1));%en grados
theta(5)=x(5);%en grados
```

```
r=sqrt(x(1)^2+x(2)^2)-a1-d5*cos(pitch);

h=x(3)-d1-d5*sin(pitch);

gamma=atan2(h,r);%en radianes

b=sqrt(h^2+r^2);

alpha=atan2(sqrt(4*a2^2-b^2),b);%ien radianes

theta(2)=-(180/pi)*(alpha+gamma);%en grados

theta(3)=(360/pi)*alpha;%en grados

theta(4)=-x(4)-theta(2)-theta(3);%en grados

theta=theta(:)';
```

Función x=ScorDeg2X(theta)

```
% FUNCTION XYZPR=ScorDeg2X(BSEPR)

% XYZPR=ScorDeg2X(BSEPR) calcular el vector de posición efecto final, XYZPR,

% (centímetros y grados)como una función del vector de ángulo Denavit-Hardenburg,

% BSEPR- 1x5 vector de cinco ángulos articulares D-H en grados.

function x=ScorDeg2X(theta)

d1=34.9;

a1=1.6;

a2=22.1;

a3=22.1;

d5=14.5125;

x(5)=theta(5);%en grados
```

```
x(4)=-theta(2)-theta(3)-theta(4);%en grados

pitch=(pi/180)*x(4);%en radianes

tr=(pi/180)*theta;%en radianes

x(3)=d1-a2*sin(tr(2))-a3*sin(tr(2)+tr(3))+d5*sin(pitch);

r=a1+a2*cos(tr(2))+a3*cos(tr(2)+tr(3))+d5*cos(pitch);

x(2)=r*sin(tr(1));

x(1)=r*cos(tr(1));

x=x(:)';
```

II.1.3. Arquitectura dirigida por eventos[2]

Este patrón arquitectónico puede ser aplicado por el diseño e implementación de aplicaciones y sistemas que transmitan eventos entro componentes. Un sistema dirigido por eventos está compuesto típicamente de emisores de eventos (o agentes) y consumidores de eventos.

La arquitectura dirigida por eventos puede complementar la arquitectura orientada a servicios porque los servicios pueden ser activados por disparadores que se encuentran en eventos entrantes. Este paradigma es particularmente útil cuando el consumidor no proporciona algún contenedor ejecutivo propio.

Un evento puede estar hecho de dos partes, el encabezado de evento y el cuerpo de evento. El encabezado de evento puede incluir información como el nombre del evento, fecha y hora para el evento, y el tipo de evento. El texto del evento es la parte que describe lo que ha ocurrido en realidad. El cuerpo del evento no debe ser confundido con el patrón o la lógica que se puede aplicar en reacción al evento en si..

[2] http://es.wikipedia.org/wiki/Arquitectura_dirigida_por_eventos

II.1.3.1. Concepto

La ingeniería de requisitos facilita el mecanismo apropiado para comprender lo que quiere el actor interesado, analizando necesidades.

II.1.3.2. Requerimientos funcionales

Los datos ingresados deberán ser validados de acuerdo a los dominios indicados en el diccionario de datos al momento de ser ingresados.

- Requisito 01
    - Ingreso de grados numéricos o decimales con un rango de -30 a 35 en el grado de libertad A.
- Requisito 02
    - Ingreso de grados numéricos enteros o decimales con el rango de 5 a 45 en el grado de libertad B.
- Requisito 03
    - Ingreso de grados numéricos enteros o decimales con un rango de -55 a 10 para el grado de libertad C.
- Requisito 04
    - Ingreso de grados numéricos enteros o decimales con un rango de -90 a 90 para el giro de las pinzas.
- Requisito 05
    - El sistema debe de contar con un mecanismo de validación para que los datos ingresados por el operador no sean datos erróneos o no válidos.

II.1.3.3. Requerimientos no funcionales

- Requisitos de rendimiento.- será una única terminal con un único usuario que esté conectado, las peticiones hacia el sistema deben realizarse en menos de 3 segundos.
- Seguridad.- los campos editables de grados no aceptarán ingresos de cifras numéricas que estén fuera del rango definido ni Strings que se intenten ingresar.
- Fiabilidad.- las coordenadas ingresadas deberán de ser procesados exitosamente, mostrando el movimiento en tiempo real del brazo robot.
- Disponibilidad.- el sistema deberá proveer tolerancia a fallos garantizando una disponibilidad alta.
- Mantenibilidad.- el mantenimiento del sistema debe ser diario.
- Portabilidad.- todos los modelos deberán ser independientes de cualquier plataforma, la totalidad del código desarrollado deberá ser compatible con las siguientes plataformas.
    - Matlab R2012a.
    - Windows 7 32/64 bits.

II.1.3.4. Modelo del negocio

A continuación se indican y describen cada uno de los artefactos que sean generados y utilizados por el proyecto y que constituyen los entregables. Esta lista constituye la configuración de RUP[3](rational unified process) desde la perspectiva de artefactos y que se propone para este proyecto.

Es preciso destacar que de acuerdo a la metodología de RUP (y todo proceso iterativo e incremental), todos los artefactos son objeto de modificaciones a lo largo del proceso de desarrollo, con

[3] *RUP.* (s.f.). Obtenido de http://www.monografias.com/trabajos-pdf4/metodologia-rup-una-puno/metodologia-rup-una-puno.shtml

lo cual, solo al termino del proceso podríamos tener una versión definitiva y completa de cada uno de ellos. Sin embargo, el resulto de cada iteración y los hitos del proyecto están enfocados a conseguir un cierto grado de completitud y estabilidad de los artefactos. Esto será indicado más adelante cuando se presenten los objetivos de cada iteración.

El modelo del negocio describe en detalle como el negocio trabaja internamente para llevar a cabo las funciones que ejecuta. Este modelo es visto desde la perspectiva de los actores externos permitiendo situar el sistema en el contexto organizacional haciendo énfasis en los objetivos en este ámbito. Este modelo se representa con el diagrama de casos de uso, usando estereotipos específicos para ello utilizando UML[4] (Modelo de Lenguaje Unificado).

II.2.Análisis y diseño

II.2.1. Requerimientos funcionales (módulos del sistema)

II.2.1.1. VRBUILD2

VRBUILD2 es la aplicación donde se diseñó la simulación del brazo robot, es un tool kit que viene dentro de matlab ubicada en MATLAB\R2012a\toolbox\sl3d\vrealm\program.

II.2.1.2. Diseño de la pinza en VRML

[4] *UML.* (s.f.). Obtenido de http://www.docirs.cl/uml.htm

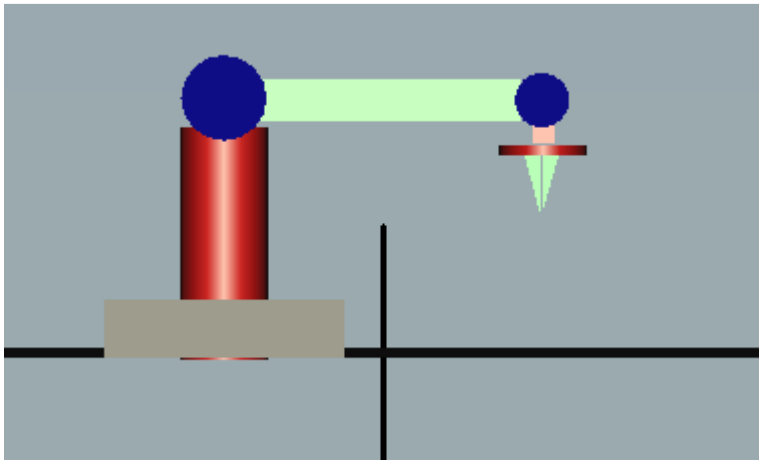


Figura 1: Diseño del brazo robot simulado.

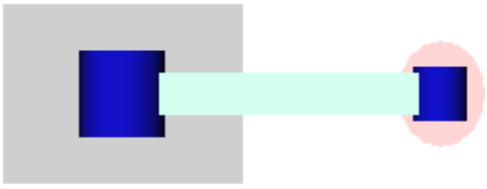


Figura 2: Vista de Arriba del brazo simulado.

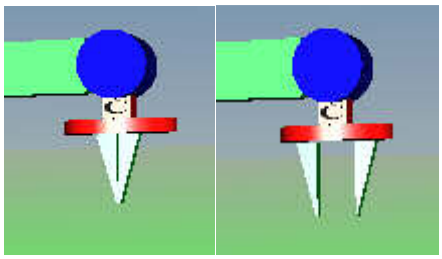


Figura 3: Pinzas cerradas y abiertas.

Construcción de las figuras en VRML

Grado 1: Figura geométrica cilindro

Centro: (0, -0.85, 0)

Rotación: (0, 0, 1, R)

Escala: (1, 1.2480, 1)

Orientación de escala: (0, 0, 1, 0)

Traslación: (0, -0.0138, 0)

Geometría del Cilindro

Radio: 0.4

Altura: 1.7

Valor de R entre -180 y 180

Objeto: Rotor 1

Centro: (0, 0, 0)

Rotación: (1, 0, 0, 90.0002)

Escala: (1, 1, 1)

Orientación de escala: (0, 0, 1, 0)

Traslación: (0, 1.52, 0)

Geometría del Cilindro

Gardo 2: Figura geométrica Caja (Box)

Centro: (-1.2040, 0, 0)

Rotación: (0, 0, 1, R)

Escala: (1, 1, 1)

Orientación de escala: (0, 0, 1, 0)

Traslación: (1.2040, 1.5, 0)

Geometría de Caja

Tamaño: (3, 0.4, 0.4)

Objeto: Rotor 2

Centro: (0, 0, 0)

Rotación: (0, 0, 1, 90.0002)

Escala: (1, 1, 1)

Orientación de escala: (0, 0, 1, 0)

Traslación: (0, 1.52, 0)

Geometría de Cilindro

Radio: 0.39

Altura: 0.8

Objeto: Rotor 4

Centro: (0, 0, 0)

Rotación: (1, 0, 0, 90.0002)

Escala: (1, 1, 1)

Orientación de escala: (0, 0, 1, 0)

Traslación: (1.6950, 0, 0)

Geometría de Cilindro

Radio: 0.25

Altura: 0.5

Objeto C: Figura geométrica Caja (Box)

Centro: (0, 0.288, 0)

Rotación: (0, 0, 1, R)

Escala: (1, 0.5, 1.2)

Orientación de escala: (0, 0, 1, 0)

Traslación: (1.7040, -0.288, 0)

Geometría de Caja

Objeto Pinza: Figura geométrica Cilindro

Centro: (-1.2040, 0, 0)

Rotación: (0, 1, 0, R)

Escala: (1, 1, 1)

Orientación de escala: (0, 0, 1, 0)

Traslación: (0, -0.6160, 0)

Objeto: Garra_1

Centro: (0, 0, 0)

Rotación: (-1, 0, 0, 88.20)

Escala: (0.2, 0.1, 0.5)

Orientación de escala: (0, 0, 1, 0)

Traslación: (0, -0.6160, 0.024)

Donde T va de -0.19 a -0.448

Objeto: Garra_1_1

Centro: (0, 0, 0)

Rotación: (-1, 0, 0, 88.20)

Escala: (0.2, 0.1, 0.5)

Orientación de escala: (0, 0, 1, 0)

Traslación: (0, -0.6160, 0.024)

Donde T va de 0.22 a 0.448

II.2.1.3. Gestionar Movimiento del Brazo Robot

Se ingresarán los grados de libertad que serán traducidos como movimientos para el brazo robot, en donde además se dispondrá de

otras funcionalidades, llevar a home[5] el brazo robot, guardar posiciones del brazo, correr la secuencia de movimientos anteriormente guardados, limpiar secuencias guardadas.

II.2.1.4. Gestionar Pinzas Robot

Se controla el giro de la pinza como también en que momento abrirla o cerrarla.

[5] Posición de trabajo del brazo robot en el espacio [X Y Z]

II.3.Diagramas de casos de uso

Los diagramas de Casos de Uso definidos para el Sistema informático desarrollado, se muestra a continuación:

Elementos de los Diagramas de Casos de Uso	*Conectores de los Diagramas de Casos de Uso*
Actor	Usa
Caso de Uso	Asociar
Colaboración	Generalizar
Limite	Incluir
Paquete	Extender
	Realizar
	Dependencia
	Trazar

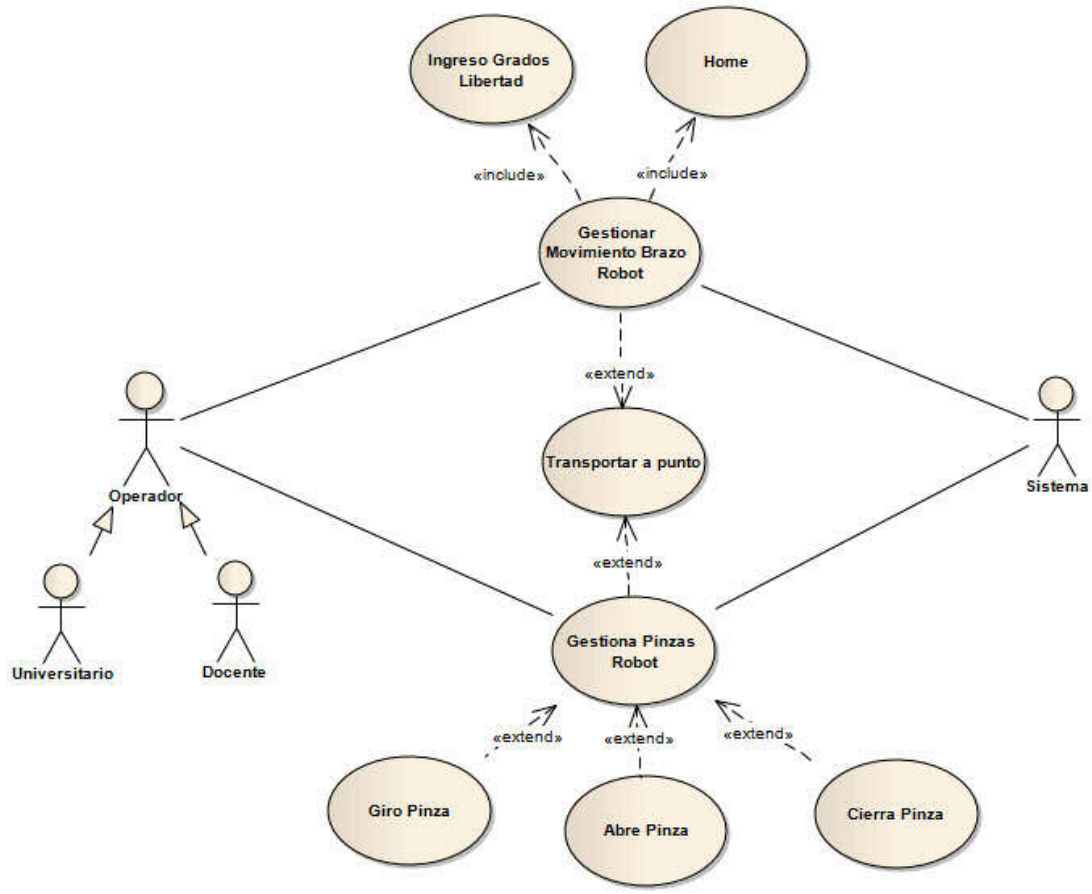
Fuente: Guía de Usuario de Enterprise Architect

Tabla 2: Etiquetas de los Diagramas de Casos de uso

II.4.Modelo de Casos de Uso del Negocio

Es un modelo de las funciones de negocio vistas desde la perspectiva de los actores externos. Permite situar al sistema en el contexto organizacional haciendo énfasis en los objetivos en este ámbito. Este modelo se representa con un Diagrama de Casos de Uso, aplicando estereotipos específicos para cada uno.

II.4.1. Modelado de Casos de Uso del Negocio: Gestionar Movimiento y pinzas del brazo robot



Fuente: Elaboración propia.

Figura 4: Diagrama de c.u. de negocio Gestionar movimiento y pinzas del brazo robot.

Descripción de casos de uso del negocio

MODULO	Gestión Movimiento Brazo Robot
TIPO	Primario.

ACTORES INVOLUGRADOS	Universitario, Docente.
DESCRIPCION	El diagrama de casos de uso “gestión movimiento brazo robot”, ejecutar varios movimientos del brazo, empezando desde el ingreso de los grados de libertad, llevar a su posición de trabajo del brazo robot (Home), luego el ingreso de grados de libertad controlando los movimientos que deberá hacer de acuerdo a los límites de movientes expuestos en la aplicación.
MODULO	Gestión Pinzas Robot
TIPO	Primario.
ACTORES INVOLUGRADOS	Universitario, Docente.
DESCRIPCION	El diagrama de casos de uso “gestión pinzas robot”, controla la pinza del robot, desde el giro de la pinza hasta la apertura y cierre de la misma.

Fuente: Elaboración propia.

Tabla 3: Descripción de c.u del negocio.

II.5.Descripción de módulos

II.5.1. Modelo de Casos de uso del Negocio

II.5.1.1. Descripción de actores del negocio

Docente: Son personas encargadas de dar cátedra en la carrera de Ingeniería Informática con título de licenciatura mínimamente en ingeniería informática o ramas afines. Será encargado de dar materia 9º semestre de Robótica INF-511, o de ser encargado del laboratorio de robótica.

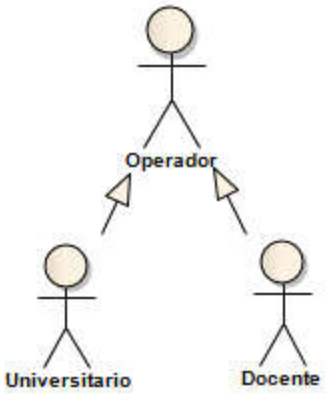
Universitario: Es la población universitaria de U.A.J.M.S[6] de la carrera de Ingeniería Informática que curse desde 7º hasta 10º semestre.

II.6.Modelo de Casos de Uso

El modelo de casos de uso presenta las funciones del sistema y los actores que hacen uso de ellas. Se representan mediante Diagramas de Casos de Uso.

[6] Juan Misael Saracho

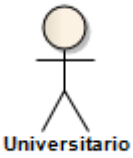
II.6.1. Identificación y descripción de actores



Fuente: Elaboración propia.

Figura 5: Identificación de Actores

II.6.2. Identificación y descripción de actores

	El universitario ingresará los grados de libertad, guardará posiciones, correrá secuencias operará la pinza.
	El docente ingresará los grados de libertad, guardará posiciones, correrá secuencias operará la pinza.

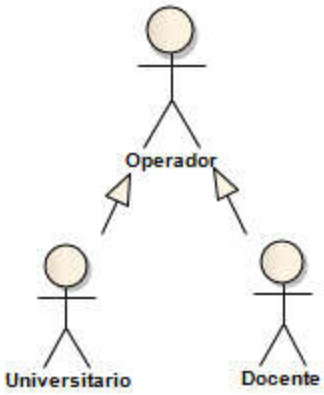
Fuente: Elaboración propia

Figura 6: Descripción de Actores del sistema

II.7.Diagramas de casos de uso

Explica gráficamente un conjunto de casos de uso de un sistema, los actores y la relación entre estos y los casos de uso. Describiendo lo que hace el sistema para cada tipo de usuario, es decir la forma en que los actores usan el sistema se presenta con un caso de uso.

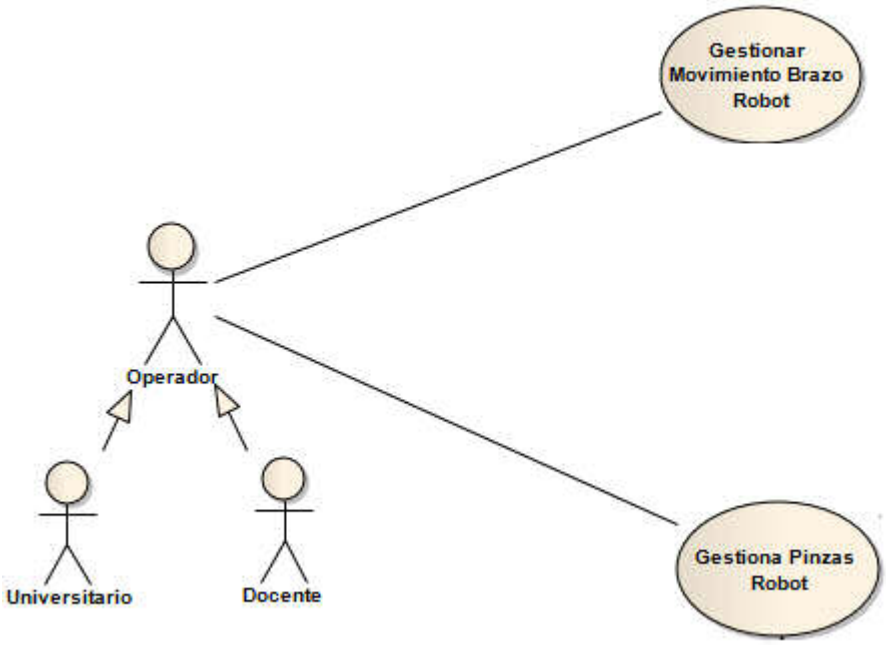
II.7.1. **Identificación de Actores –** Usuarios del Sistema



Fuente: Elaboración propia.

Figura 7: Identificación de actores – usuarios del sistema

II.7.2. **Diagramas de Casos de Uso:** Caso de Uso General

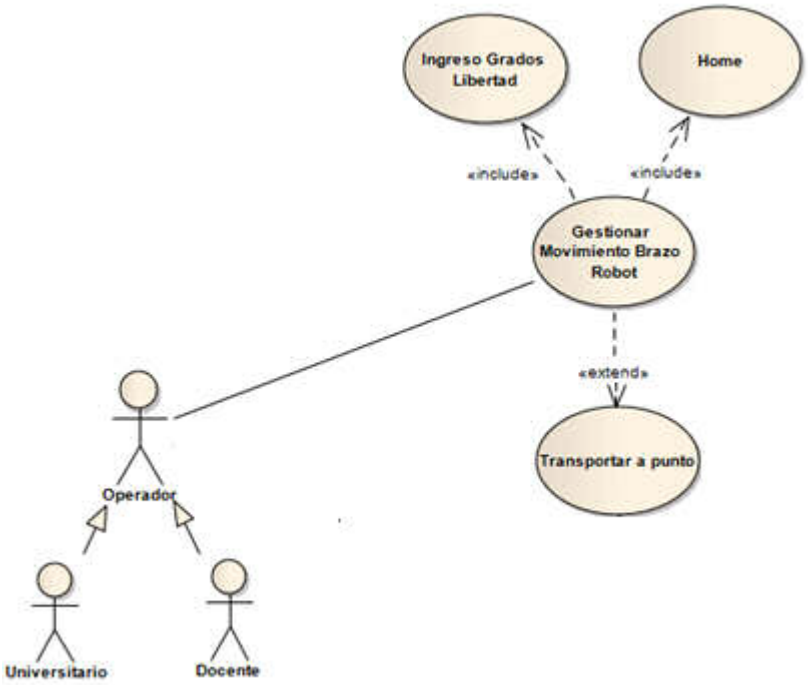


Fuente: Elaboración propia.

Figura 8: Diagrama de c.u. general.

II.8.Diagrama de casos de uso específicos

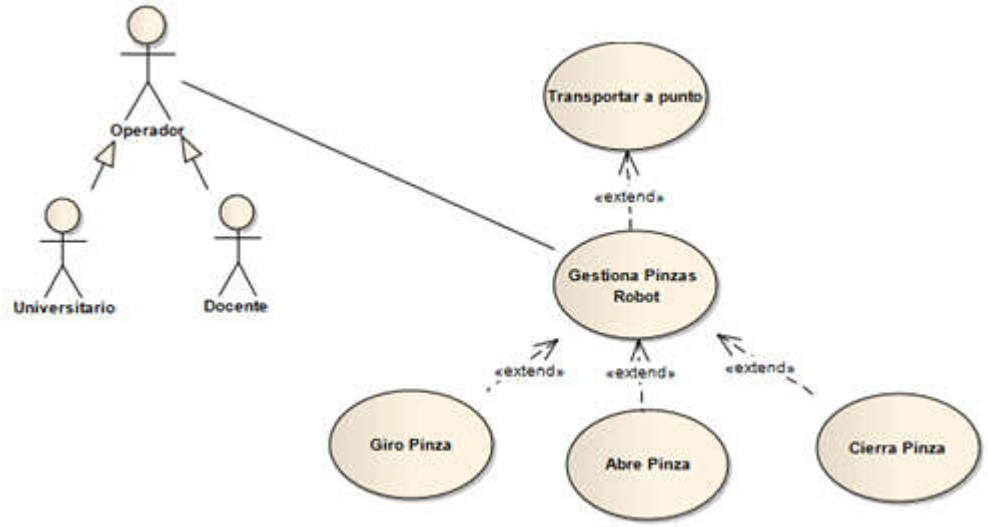
II.8.1. **Modelo de Casos de Uso:** Gestión Movimiento Brazo Robot.



Fuente: Elaboración propia.

Figura 9: Diagrama de c.u. Gestión Movimiento Brazo.

II.8.2. **Modelo de Casos de Uso:** Gestión Pinzas Robot.



Fuente: Elaboración propia.

Figura 10: Diagrama de c.u. Gestión Pinzas Robot.

II.9.Descripción de los casos de uso

La especificación de casos de uso es una descripción detallada de los casos de uso del sistema, tiene como propósito comprender y describir espáticamente cada caso de uso del sistema. Tiene como alcance describe los procesos internos de los casos de uso y los flujos de cada caso de uso según requerimiento de la organización.

II.9.1. Descripción de Casos de Uso Gestionar movimiento Brazo Robot

II.9.1.1. Descripción de Caso de Uso: Ingreso grados de libertad

Caso de uso:	**Ingreso grados de libertad**
Actores:	Operador
Tipo	Caso de Uso Secundario
Propósito:	Ingreso de grados de libertad al sistema

Descripción:

Está a la espera de datos para ser ingresados y así ejecutar los movimientos que sean especificados por el operador.

Pre-condición:	Que se haya colocado valores válidos como grados de libertad del brazo.
Post-condición:	

Flujo principal:

1. El operador selecciona cualquiera de los cuadros de texto debajo de la etiqueta “Edit Grados” de la pantalla principal. 2. El sistema ejecuta la petición del operador traduciéndolo en movimiento del brazo robot, tanto físicamente y en la simulación.

Flujo alternativo
Excepciones:
Mensaje de error 1 “El rango incorrecto [--,--]” Mensaje de error 2 “El valor debe ser numérico”

Fuente: Elaboración propia.

Tabla 4: Descripción de casos de uso Ingreso grados de libertad

II.9.1.2. Descripción de Casos de Uso: Home

Caso de uso:	**Home**
Actores:	Operador
Tipo	Caso de Uso Secundario
Propósito:	Coloca al brazo robot en su posición de trabajo “Home”.
Descripción:	
Se coloca en su posición de trabajo del brazo robot, y también va revisando eje por eje del brazo que este operacional.	
Pre-condición:	Que haya posiciones en la tabla de secuencia, o se haya movido el brazo después de su activación colocándolo en algún punto.
Post-condición:	

Flujo principal:
1. El operador ingresa el parámetro dentro del cuadro de texto debajo del rotulo "Edit Grados" de la interfaz principal. 2. El sistema ejecuta la petición girando las pinzas del brazo robot.
Flujo alternativo
Excepciones:

Fuente: Elaboración propia.

Tabla 5: Descripción de casos de Uso Home

II.9.2. Descripción de los casos de uso Gestiona Pinzas Robot

II.9.2.1. Descripción de Casos de Uso: Giro Pinza

Caso de uso:	**Giro Pinza**
Actores:	Operador
Tipo	Caso de Uso Secundario
Propósito:	Girar pinzas del brazo robot.
Descripción:	
Se girará la pinza de acuerdo al parámetro que se ingresen.	
Pre-condición:	
Post-condición:	

Flujo principal:
1. El operador presiona el botón “Giro Pinza” de la interfaz principal. 2. El sistema ejecuta la petición girando la pinza del brazo robot.
Flujo alternativo
Excepciones:
Mensaje de error 1 “El rango incorrecto [--,--]” Mensaje de error 2 “El valor debe ser numérico”

Fuente: Elaboración propia.

Tabla 6: Descripción de casos de uso Giro Pinza

II.9.2.2. Descripción de Casos de Uso: Abre Pinza

Caso de uso:	**Abre Pinza**
Actores:	Operador
Tipo	Caso de Uso Secundario
Propósito:	Abre las pinzas del brazo robot.
Descripción:	
Se abre la pinza al presionar el botón “Abrir pinza”.	
Pre-condición:	Que las pinzas estén cerradas
Post-condición:	

Flujo principal:
1. El operador presiona el botón “Abrir pinza” de la interfaz principal. 2. El sistema ejecuta la petición abriendo la pinza al brazo robot.
Flujo alternativo
Excepciones:

Fuente: Elaboración propia

Tabla 7: Descripción de casos de uso Abre Pinza

II.9.2.3. Descripción de Casos de Uso: Cerrar Pinza

Caso de uso:	**Cerrar Pinza**
Actores:	Operador
Tipo	Caso de Uso Secundario
Propósito:	Cerrar las pinzas del brazo robot.
Descripción:	
Se abre la pinza al presionar el botón “Abrir pinza”.	
Pre-condición:	Que las pinzas estén abiertas
Post-condición:	
Flujo principal:	

1. El operador presiona el botón “Cerrar pinza” de la interfaz principal. 2. El sistema ejecuta la petición cerrando la pinza al brazo robot.
Flujo alternativo
Excepciones:

Fuente: Elaboración propia.

Tabla 8: Descripción de casos de uso Cerrar pinza

II.9.2.4. Descripción de Casos de Uso: Transportar posición.

Caso de uso:	Transportar posición
Actores:	Operador
Tipo	Caso de Uso Secundario
Propósito:	Movimiento del brazo robot a un

	determinado punto.

Descripción:

Se moverá a un punto determinado por el operador físicamente.

Pre-condición:	Que se hayan colocado las coordenadas deseadas.
Post-condición:	

Flujo principal:

3. El operador presiona el botón “Transportar posición” de la interfaz principal. 4. El sistema ejecuta la petición transportando el brazo robot al punto valido deseado físicamente.

Flujo alternativo

Excepciones:

Fuente: Elaboración propia.

Tabla 9: Descripción de casos de Uso Transportar pinza

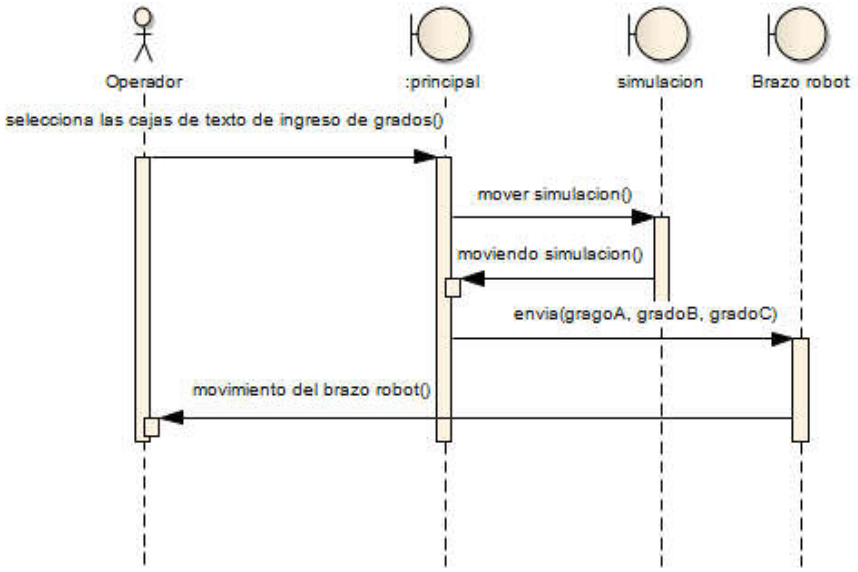
II.10. Diagramas de secuencia

Los diagramas de secuencia son dibujos, que muestran el orden y los eventos de entrada y salida que generan los actores externos descritos en los casos de uso.

Su propósito es el de comprender la estructura, dinámica e interacción de los actores y el sistema diseñado para la organización.

Su alcance es describir un escenario específico de un caso de uso, la interacción entre los objetivos del sistema y representar las interacciones entre actores y operaciones.

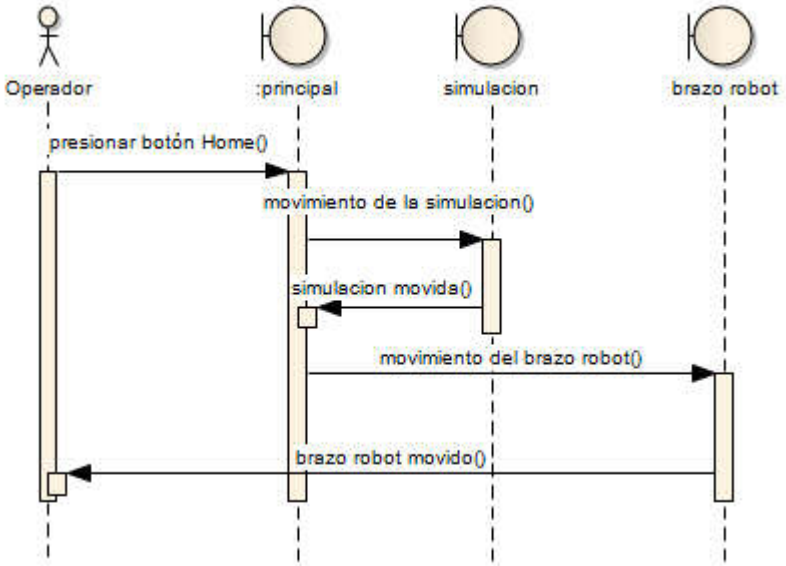
II.10.1. Diagramas de Secuencia: Ingreso grados de libertad



Fuente: Elaboración propia

Figura 11: Diagrama de Secuencia Ingreso grados de libertad

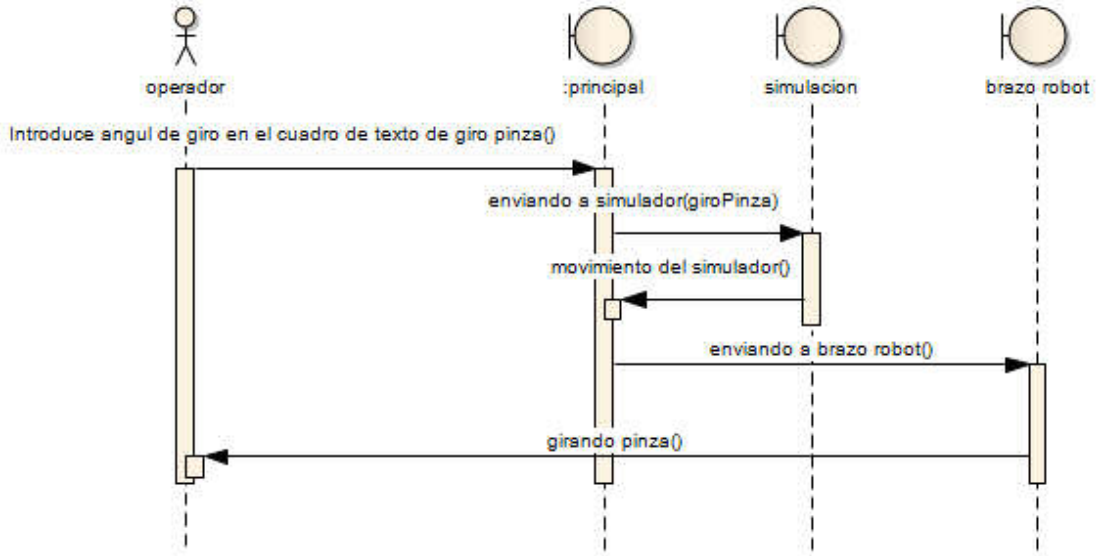
II.10.2. Diagrama de Secuencia: Home



Fuente: Elaboración propia

Figura 12: Diagrama de Secuencia Home

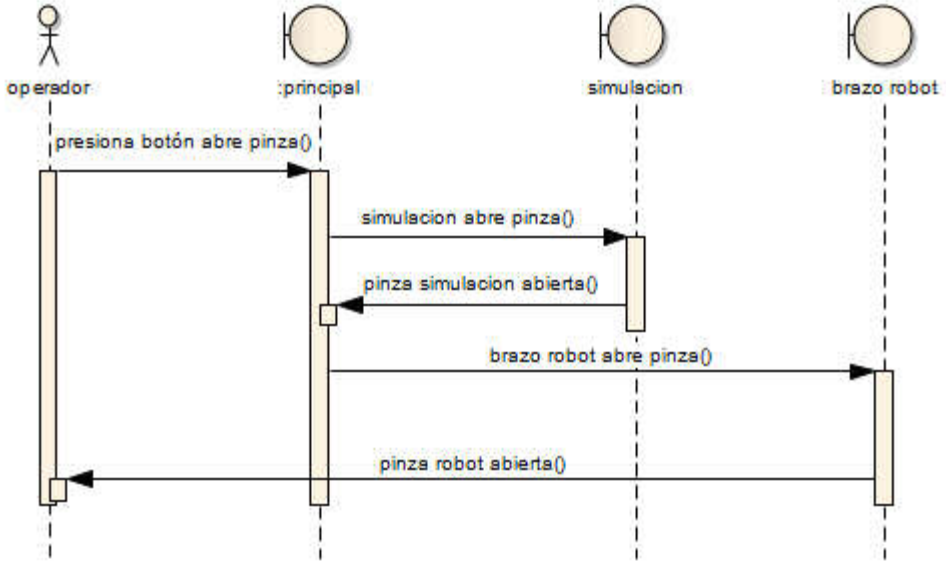
II.10.3. Diagrama de Secuencia: Giro pinza



Fuente: Elaboración propia.

Figura 13: Diagrama de Secuencia Giro pinza

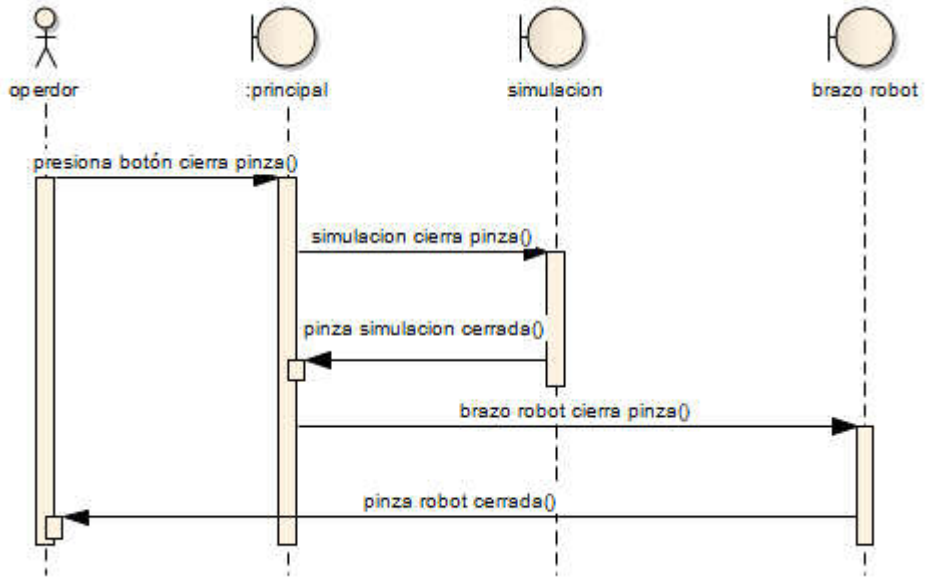
II.10.4. Diagramas de Secuencia: Abre pinza



Fuente: Elaboración propia

Figura 14: Diagrama de Secuencia: Abre pinza

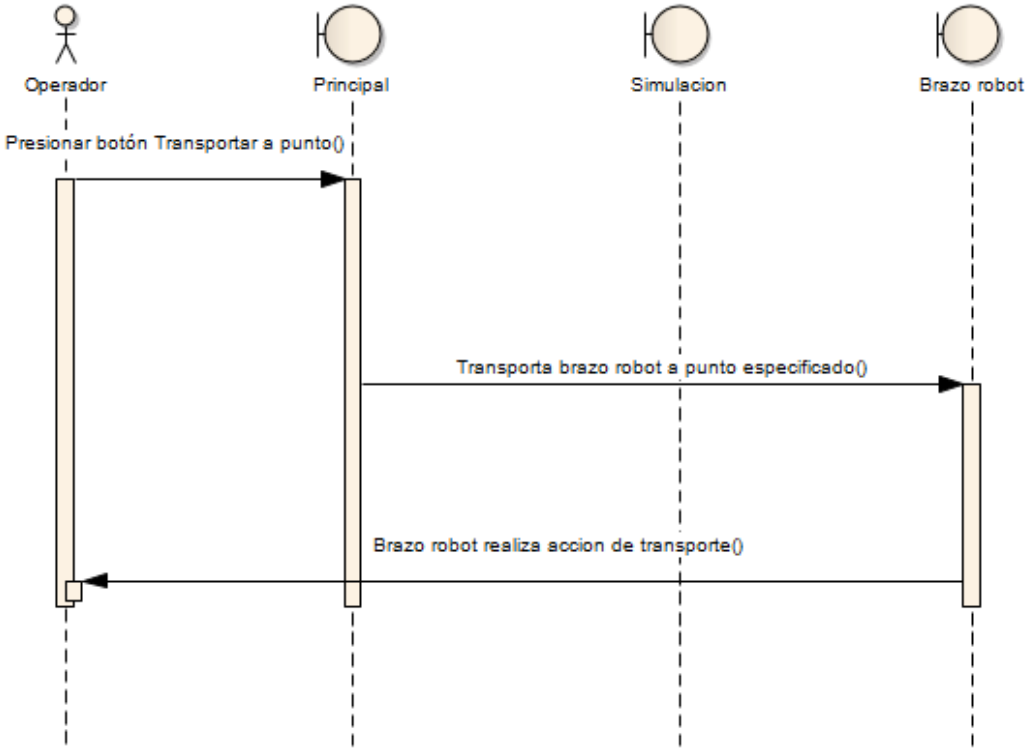
II.10.5. Diagrama de Secuencia: Cerrar pinza



Fuente: Elaboración propia

Figura 15: Diagrama de Secuencia cerrar pinza

II.10.6. Diagrama de Secuencia: Transportar a punto.



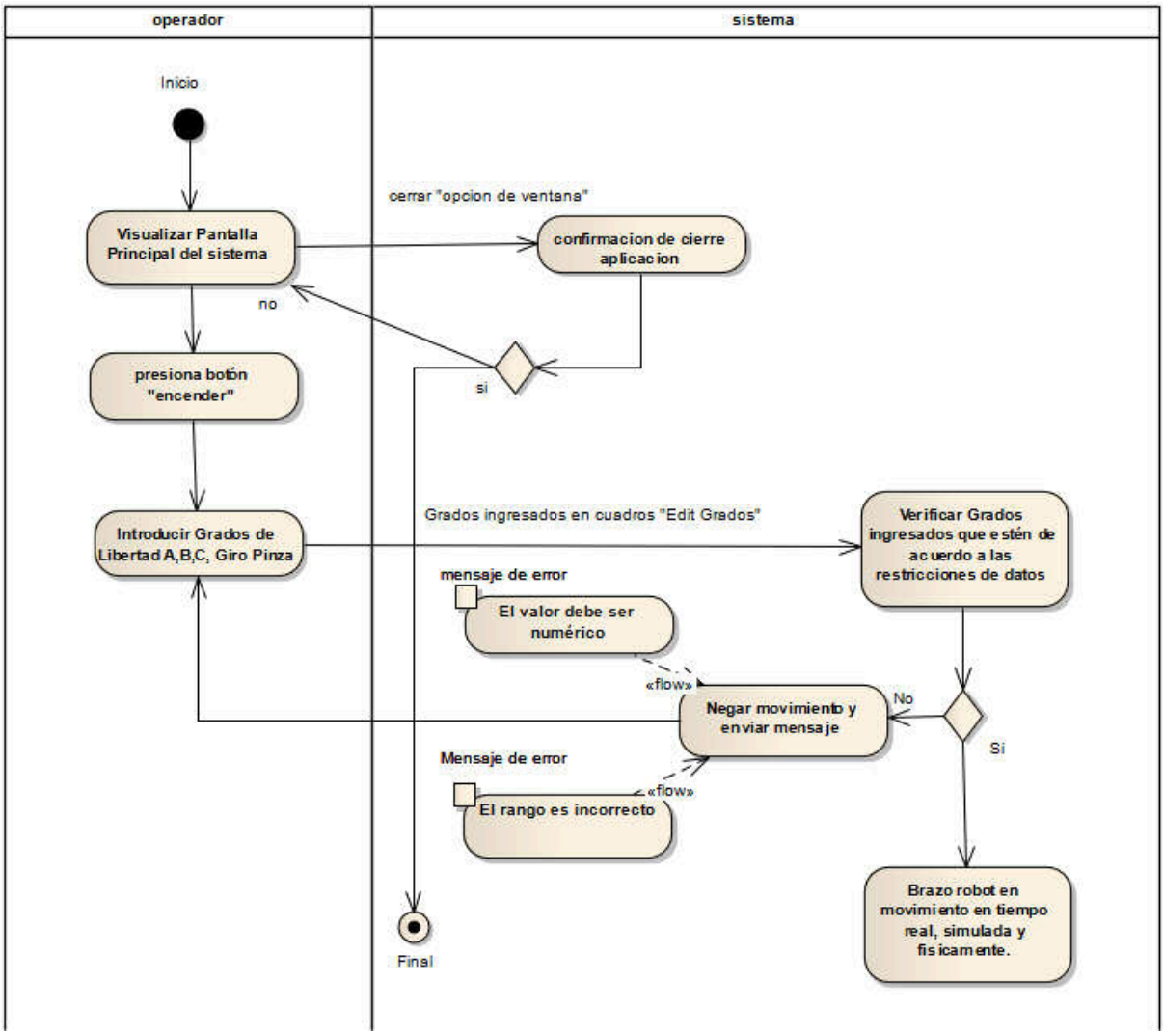
Fuente: Elaboración propia.

Figura 16: Diagrama de secuencia transportar punto

II.11. Diagramas de Actividad del Sistema

Un diagrama de actividad de UML ofrece una notación rica para representar una actividad, podría aplicarse a cualquier propósito (como para mostrar los pasos del algoritmo) pero se considera útil para visualizar los flujos de trabajo y los procesos del negocio o caso de uso.

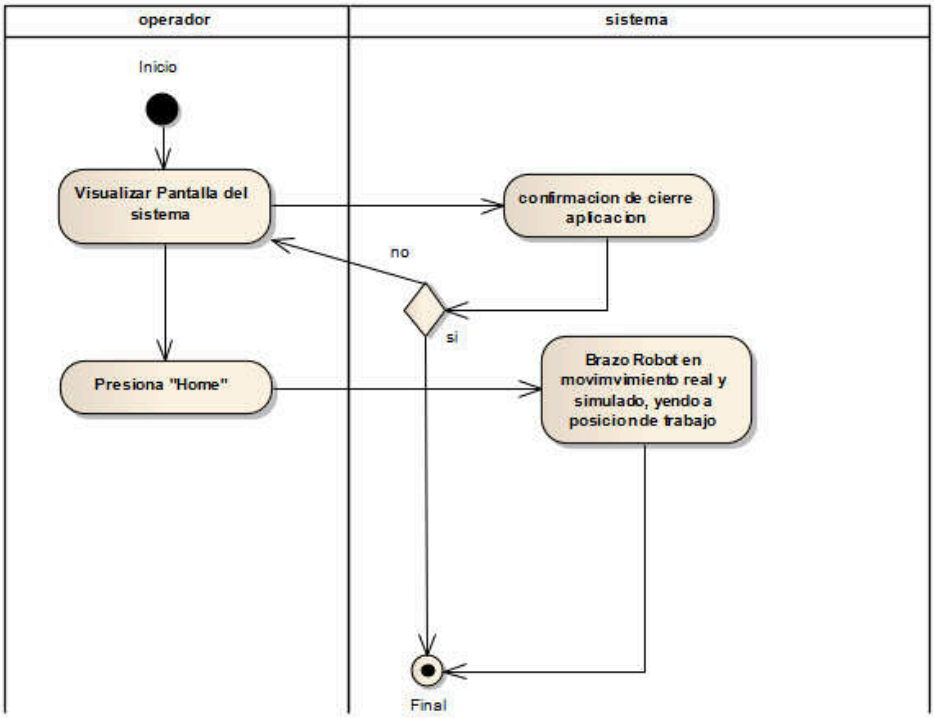
II.11.1. Diagrama de Actividades: Gestionar Movimiento Brazo Robot



Fuente: Elaboración propia

Figura 17: Diagrama de Actividades: Gestionar Movimiento Brazo Robot

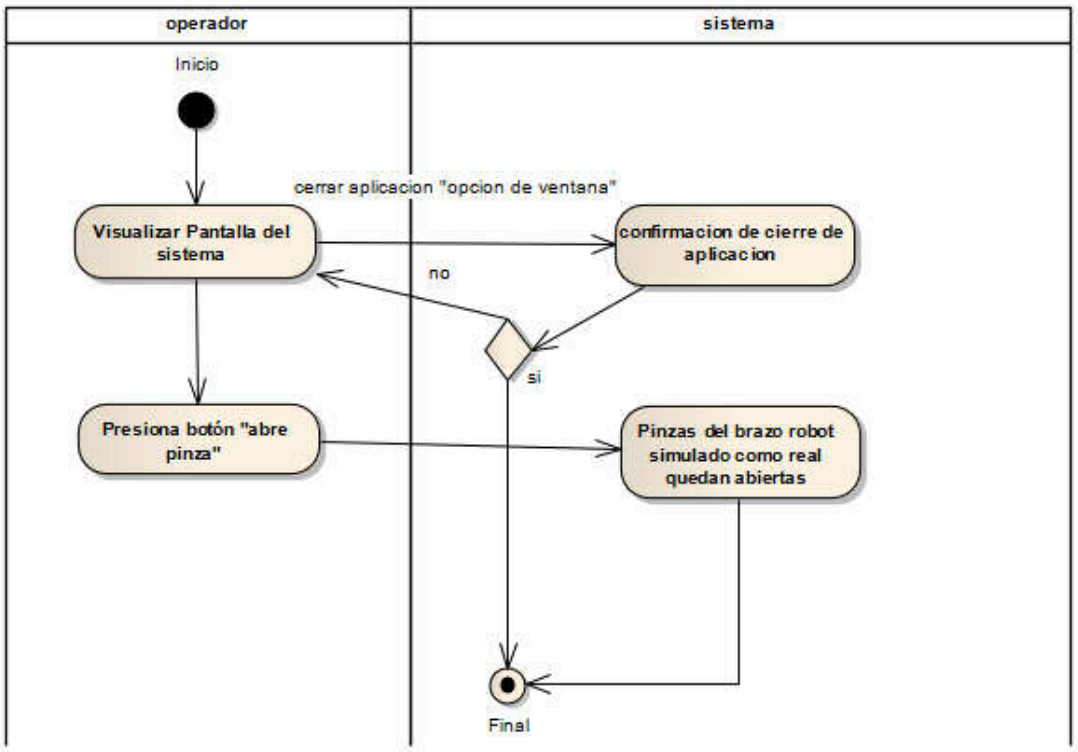
II.11.2. Diagrama de Actividades: Home



Fuente: Elaboración propia

Figura 18: Diagrama d actividades Home

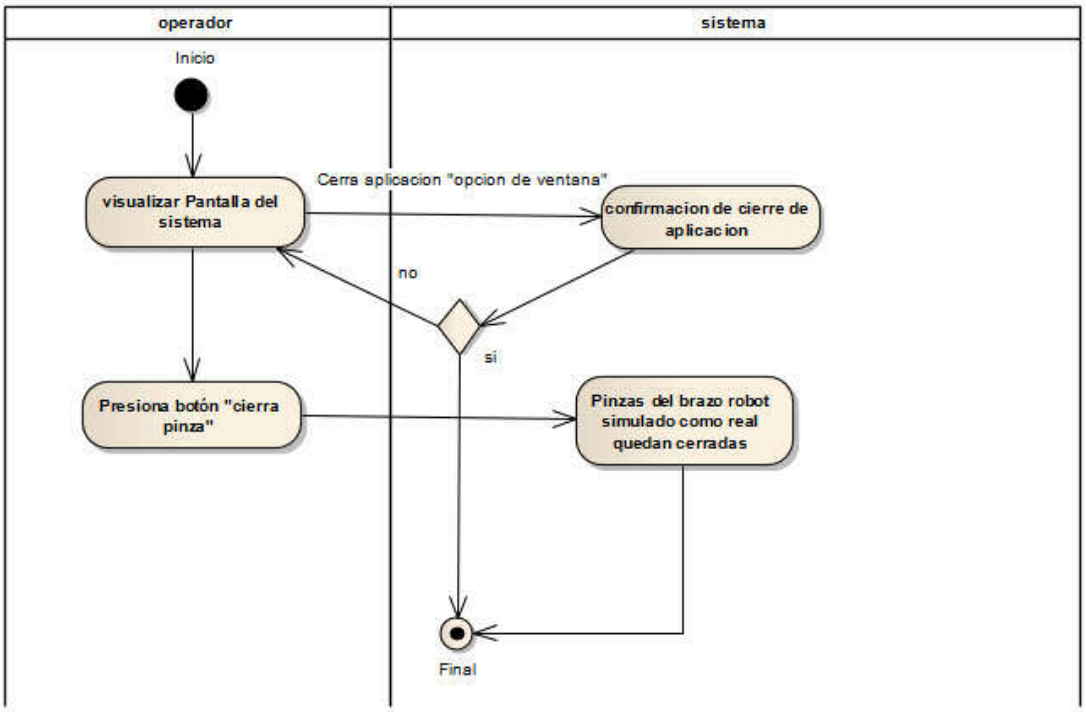
II.11.3. Diagrama de Actividades: Abre Pinza



Fuente: Elaboración propia

Figura 19: Diagrama de actividades Abre pinza

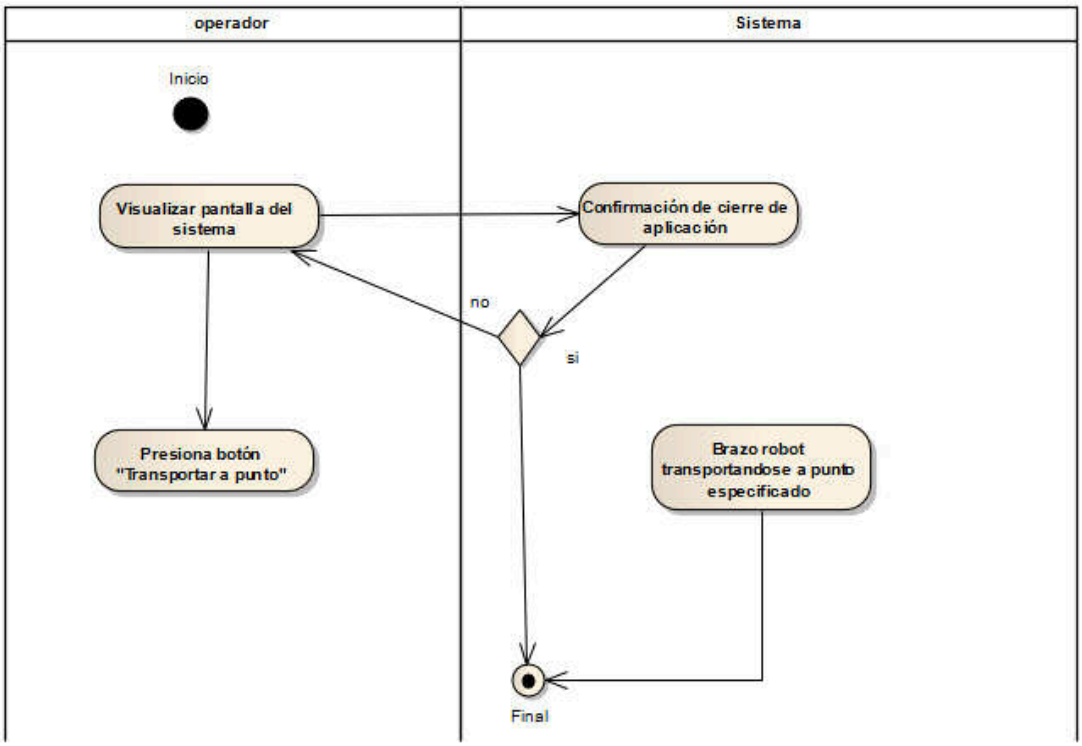
II.11.4. Diagrama de Actividades: Cerrar pinza



Fuente: Elaboración propia

Figura 20: Diagrama de Actividades Cerrar pinza

II.11.5. Diagrama de Actividades: Transportar a punto.



Fuente: Elaboración propia.

Figura 21: Diagrama de Actividades: Transportar a punto

II.12. Diagramas de Componentes

El modelo de componentes ilustra los componentes de software que se usarán para construir el sistema. Se pueden construir a partir del modelo de clase o escribir desde cero para el nuevo sistema o se puede importar de otros proyectos y de productos de terceros. Los componentes son agregados de alto nivel de las piezas de software más pequeñas y proveen un enfoque de construcción de bloques de "caja negra" para la elaboración de software.

Notación de Componentes

Un componente puede ser algo como un control, tanto un componente de la interfaz de usuario como un servidor de reglas de negocio. Los componentes se representan gráficamente[7] como se muestra a continuación:

[7] http://www.sparxsystems.com.ar/EAUserGuide/ea.html?componentdiagram.htm

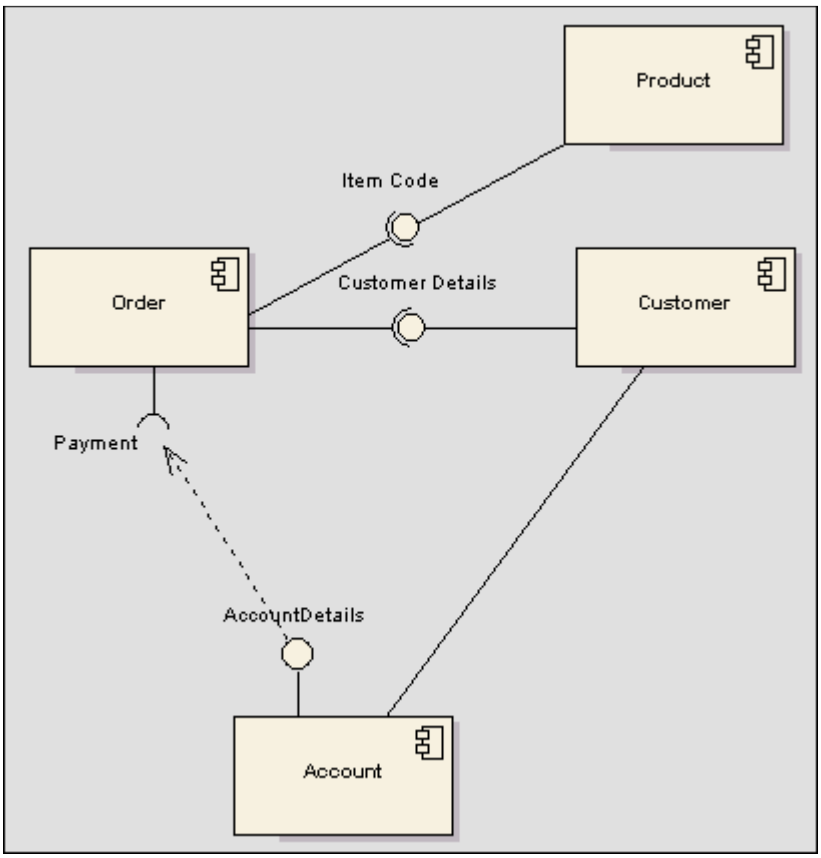
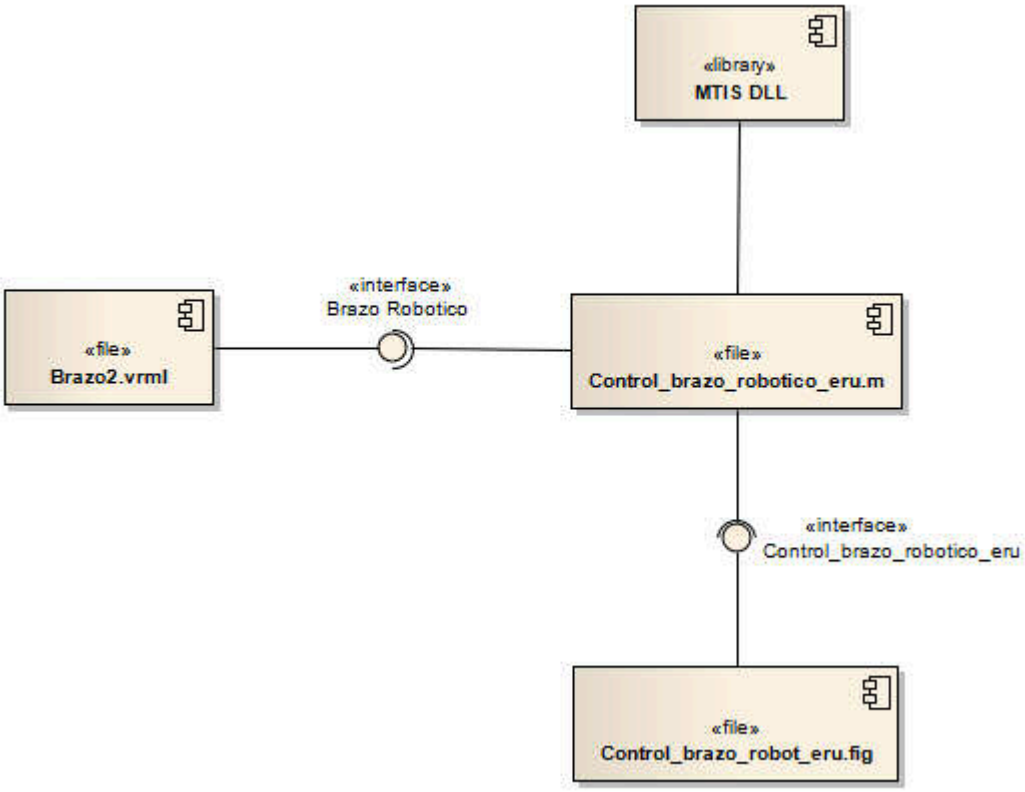


Figura 22: Diagrama de componentes ejemplo

El diagrama de componentes

El diagrama de componentes muestra la relación entre componentes de software, sus dependencias, su comunicación su ubicación y otras condiciones.

II.12.1. Diagrama de Componentes: Modelo General



Fuente: Elaboración propia

Figura 23: Diagrama de Componentes Modelo General

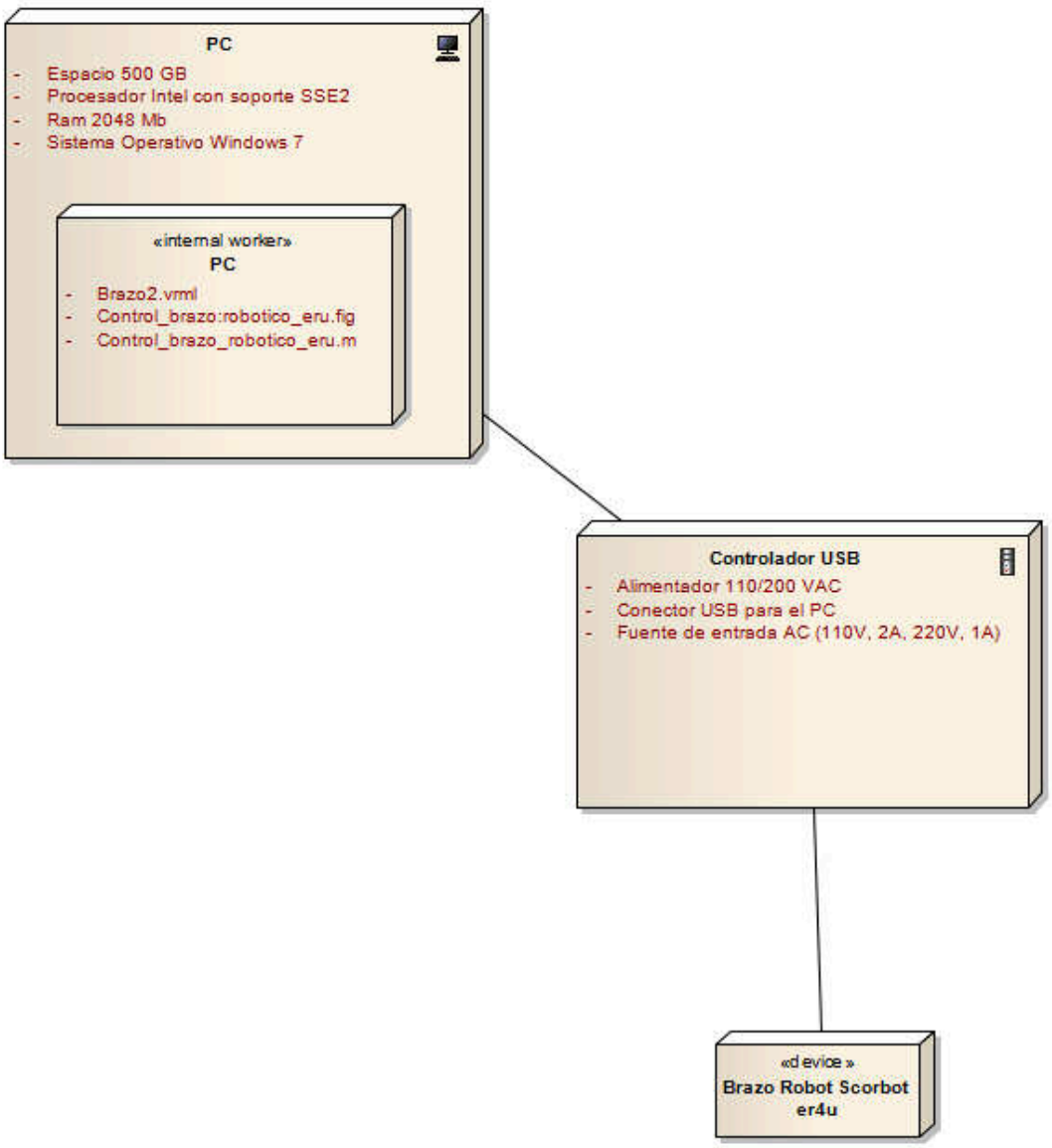
II.13. Diagrama de despliegue

El modelo físico de desplegué provee un modelo detallado de la forma en la que los componentes se despliegan a lo largo de la infraestructura del sistema. Detalla las capacidades de red, las especificaciones del servidor, los requisitos de hardware y otra información relacionada al despliegue del sistema propuesto.

Un nodo se representa como un cubo, un nodo es un elemento donde se ejecutan los componentes, representan el despliegue físico de estos componentes.

Como los componentes pueden residir en más de un nodo podemos situar el componente de forma independiente, sin que pertenezca a ningún nodo, y relacionarlo con los nodos en los que se sitúan.

II.13.1. Diagrama de Despliegue: Modelo General



Fuente: Elaboración propia

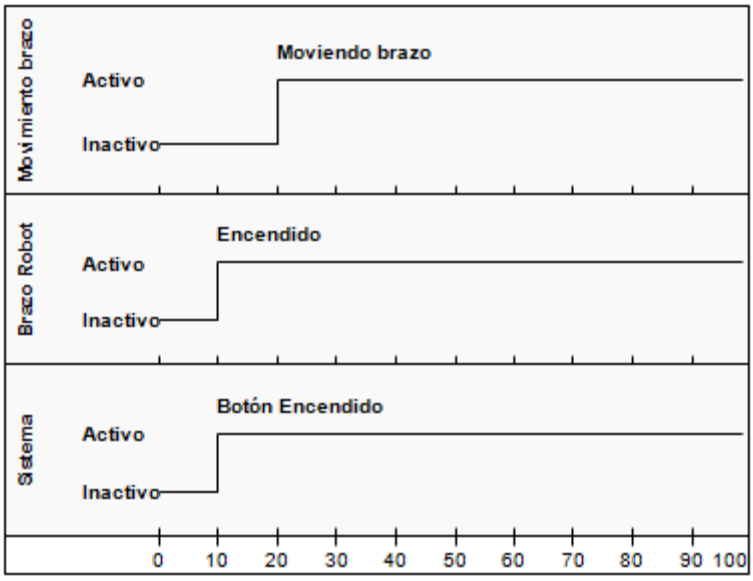
Figura 24: Diagrama de Despliegue Modelo General

II.14. Diagrama de Tiempo[8]

El diagrama de Tiempo define el comportamiento de los diferentes objetos con una escala de tiempo. Provee una representación visual de los objetos cambiando de estado e interactuando a lo largo del tiempo

Los diagramas de tiempos se pueden usar para definir componentes de software dirigidos por hardware o embebidos. Puede ser un software embebido tal como el que se usa en un sistema de inyección de combustible, un controlador de microondas, etc. También se pueden usar para especificar procesos de negocio dirigidos por tiempo.

II.14.1. Diagrama de Tiempo: Moviendo Brazo



Fuente: Elaboración propia

Figura 25: Diagrama de Tiempo Movimiento Brazo

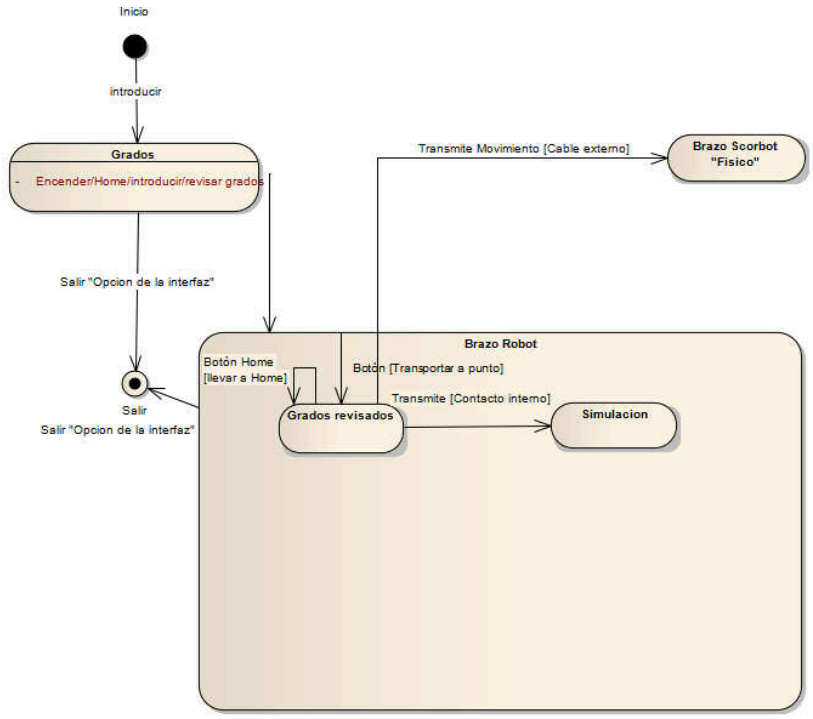
El brazo estará inactivo hasta que el sistema lanza la señal de activo.

[8] http://www.sparxsystems.com.ar/EAUserGuide/ea.html?timingdiagram.htm

II.15. Diagramas de Estado[9]

Un diagrama de Maquina de Estados ilustra como un elemento, muchas veces una clase, se puede mover entre estados que clasifican su comportamiento, de acuerdo con disparadores, guardias de restricción y otros aspectos de los diagramas de máquinas de estado que representarán y explicarán el movimiento y el comportamiento.

II.15.1. Diagrama de Estados: Modelo General



Elaboración Propia

Figura 26: Diagrama de Estados Modelo General

[9] http://www.sparxsystems.com.ar/EAUserGuide/ea.html?statediagram.htm

II.16. Prototipo de interfaces de usuario

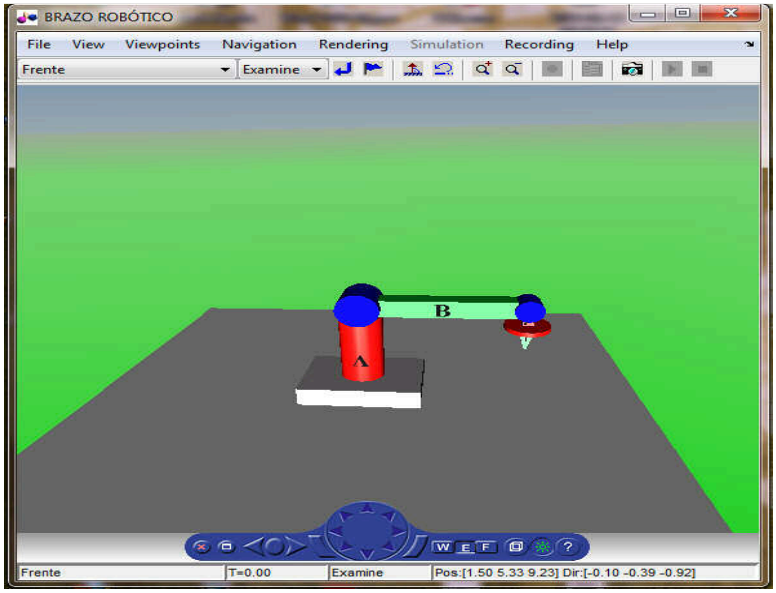
Se trata de prototipos que permiten al usuario hacerse una idea más o menos precisa de las interfaces que proveerá el sistema y así, conseguir retroalimentación de su parte respecto a los requisitos del sistema.

Estos prototipos se realizarán como: dibujo a mano en papel, dibujo con alguna herramienta gráfica o prototipos ejecutables interactivos, siguiendo ese orden de acuerdo al avance del proyecto. Solo los de este último tipo serán entregados al final de la fase de elaboración, los otros serán desechados. Asimismo, este artefacto, será desechado en la fase de construcción en la medida que el resultado de las iteraciones vayan desarrollando el proyecto final.

II.16.1. Pantalla principal del sistema más pantalla simulación

Fuente: Elaboración propia

Figura 27: Prototipo del sistema Pantalla Principal



Fuente: Elaboración propia

Figura 28: Pantalla de la simulación

En estas pantallas se hará el ingreso de datos para el movimiento del brazo robot y su posterior simulación en la última pantalla mostrada.

Botón Encendido: Botón que carga las librerías para la comunicación con el control –usb.

Botón Home: Presionándolo el brazo comienza su movimiento de revisión de ejes para colocarlo en su posición de trabajo, y listo para recibir peticiones.

Edit Grados: Compuesto de cuatro text box en los cuales se ingresar´sn los grados haciendo mover el respectivo grado de libertad que se muestra (A, B, C) y el giro de pinza haciendo mover tanto el brazo robot y la simulación.

Abre pinza: Botón que al presionarlo abre las pinzas del brazo robot mostrándose tanto en la simulación como en el brazo real.

Cierra pinza: Botón que al presionarlo cierra las pinzas del brazo robot mostrándose tanto en la simulación como en el brazo real.

Botón Transportar a punto: Botón que al presionarlo mueve el brazo robot de acuerdo a los parámetros ingresados.

II.16.2. Mensaje de error Rango

Fuente: Elaboración propia

Figura 29: Mensaje de error Rango

Mensaje que aparecerá si el usuario introduce una cantidad fuera del rango establecido en cada grado de libertad.

II.16.3. Mensaje de error Evaluación de numero

Fuente: Elaboración Propia

Figura 30: Mensaje de error Evaluación numero

Mensaje que aparecerá si el usuario introduce otro valor que no sea numérico aceptado.

II.17. Casos de prueba

Cada prueba es especificada mediante un documento que establece las condiciones de ejecución, las entradas de la prueba, y los resultados esperados.

Estos casos son aplicados como prueba de regresión en cada interacción.

Cada prueba llevará asociado un procedimiento con las instrucciones para realizarlas, y dependiendo del tipo de prueba.

II.17.1. Prueba de caja negra

Las pruebas de caja permiten al ingeniero de software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos fundamentales de un programa. La prueba de caja negra no es una alternativa a las técnicas de pruebas de caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores de los métodos de caja blanca.

Las pruebas de caja negra intentan encontrar errores de las siguientes categorías.

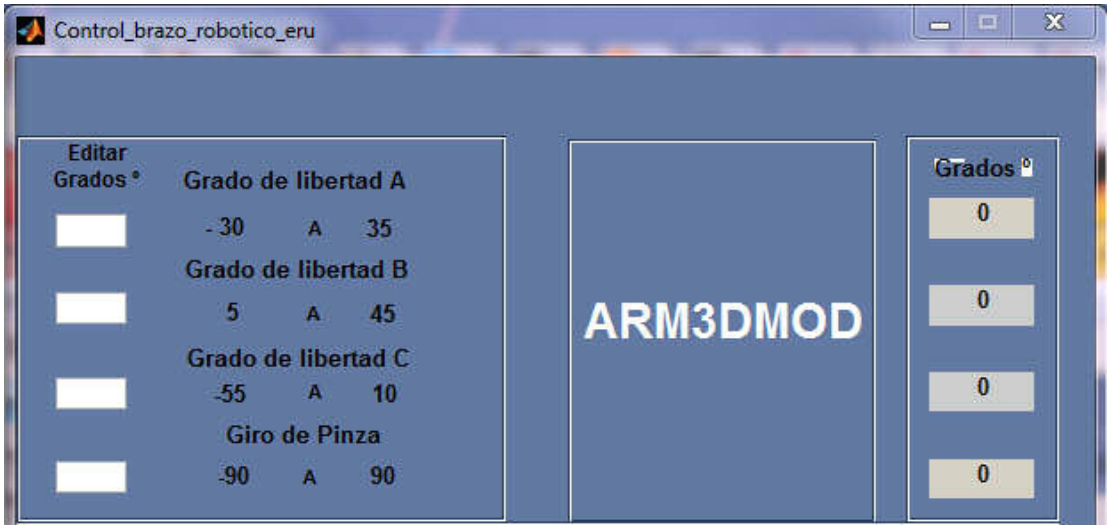
- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructura de datos o en accesos de bases de datos externos.
- Errores de rendimiento.
- Errores de inicialización y terminación.

II.17.1.1. Partición equivalente

La partición equivalente es un método de caja negra que divide la entrada de un programa en clases de datos de los que se pueden derivar casos de prueba.

La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número de casos de prueba que hay que desarrollar.

Entrada de Datos: Pantalla Principal



Fuente: Elaboración propia

Figura 31: Entrada de Datos Pantalla Principal

Paso 1: Identificación de las condiciones de entrada.

Condiciones de entrada	**C.E. Validos**	**C.E. Inválidos**
Los campos de Edit Grados º tienen.	1. Números.	2. Cualquier otra cosa.

Fuente: Elaboración propia

Tabla 10: Identificación de Datos Pantalla Principal

Paso 2: Identificación de los casos de prueba que cubran una o más clases de equivalencias.

Clases Validas

Caso	Edit Grado A, B, C, Giro pinza	Clases
C.p.1	12	1
C.p.2	12.1	1

Fuente: Elaboración propia

Tabla 11: Claves Validas

Clases Inválidas

Caso	Edit Grado A, B, C, Giro pinza	Clases
C.p.1	“”	2
C.p.2	“asd”	2
C.p.3	12.asd	2
C.p.4	Asda.12	2
C.p.5	12.””	2
C.p.6	“”.12	2

Fuente: Elaboración propia

Tabla 12: Claves Inválidas

II.17.2. Pruebas de Caja Blanca

Las pruebas de caja blanca también conocidas como pruebas de caja de cristal o pruebas estructurales, se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente. El testador escoge distintos calores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y cerciorarse de que se devuelven los valores de salida adecuados.

Al estar basadas en una implementación concreta, si esta se modifica, por regla general las pruebas también deberán rediseñarse.

Aunque las pruebas de caja blanca son aplicables a varios niveles (unidad, integración y sistema), habitualmente se aplican a las unidades de software. Su cometido es comprobar los flujos de ejecución dentro de cada unidad (función, clase, módulo, etc.) pero también pueden testear los flujos entre unidades durante la integración e incluso entre subsistemas, durante las pruebas del sistema.

A pesar de que este enfoque permite diseñar pruebas que cubran una amplia variedad de casos de prueba, podría pasar por alto partes incompletas de la especificación o requisitos faltantes, pese a garantizar las pruebas exhaustivas de todos los flujos de ejecución del código analizado.

II.17.2.1. Prueba del camino básico[10]

El método del camino básico (propuesto por McCabe) permite obtener una medida de la complejidad de un diseño procedimental, y utilizar esta medida como guía para la definición de una serie de

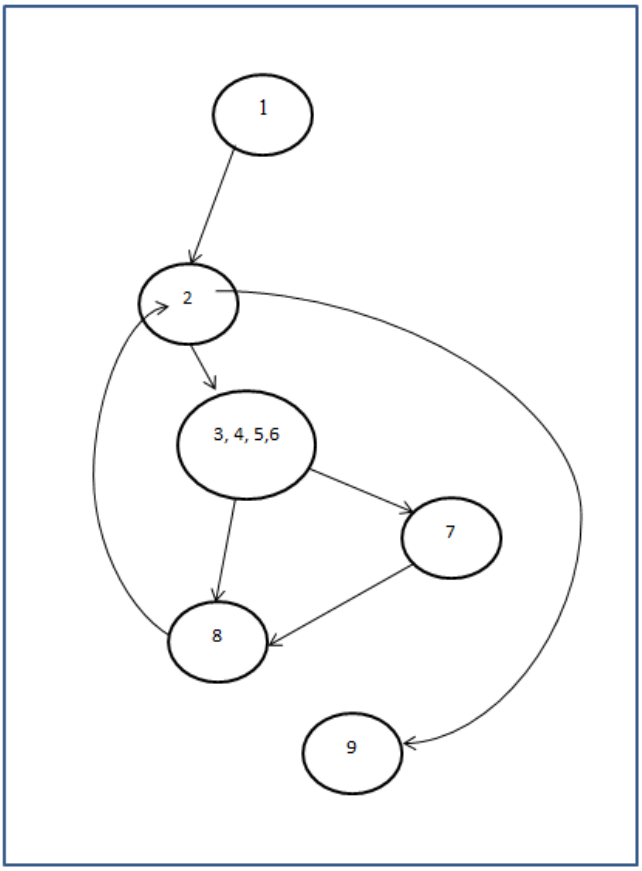
[10] http://indalog.ual.es/mtorres/LP/Prueba.pdf

caminos násicos de ejecución, desenando casos de prueba que garantices que cada camino se ejecute al menos una vez. A continuación se realizará las pruebas de camino básico al proyecto en desarrollo “Control del brazo robot scorbot er4u mediante matlab.” A las funciones más relevantes del proyecto de la interfaz principal.

II.17.2.2. Prueba de camino básico: Función Cierra_Pinza_Callback

```
function Cierra_Pinza_Callback(hObject, eventdata, handles)
global  world i
if(i==0.19)
   return
end
for i=0.4480:-0.001:0.19
Garra=vrnode(world,'Garra1');%cargando nodo de VRML
Garra.translation=[-i -0.6160 0.0240];
Garra2=vrnode(world,'Garra1_1');
   if (i>0.22)
      Garra2.translation=[i -0.6160 0.0240];
   end
vrdrawnow;
drawnow
end
```

Diagrama de Flujo de Control



Fuente: Elaboración propia

Figura 32: Diagrama de Flujo de Control Función Cierra Pinza

Complejidad ciclomatica

G(s) = aristas – nodos + 2 = 7 – 6 + 2 = 3

La complejidad ciclomatica del sistema es de 3 caminos de prueba.

Camino 1:	1,2,9
Camino 2:	1,2,3,4,5,6,7,8,2,9
Camino 3:	1,2,3,4,5,6,8,2,9

Fuente: Elaboración propia

Tabla 13: Caminos Función Cierra Pinza

II.17.2.3. Pruebas de camino básico: Función Aabre_pinza_Callback

```
function Aabre_pinza_Callback(hObject, eventdata, handles)
global  world i
if(i==0.4480)
   return
end
for i=0.19:0.001:0.4480
Garra=vrnode(world,'Garra1');%cargando nodo de VRML
Garra.translation=[-i -0.6160 0.0240];
```

```
Garra2=vrnode(world,'Garra1_1');
   if (i>0.22)
      Garra2.translation=[i -0.6160 0.0240];
   end
vrdrawnow;
drawnow
end
```

Diagrama de Flujo de Control

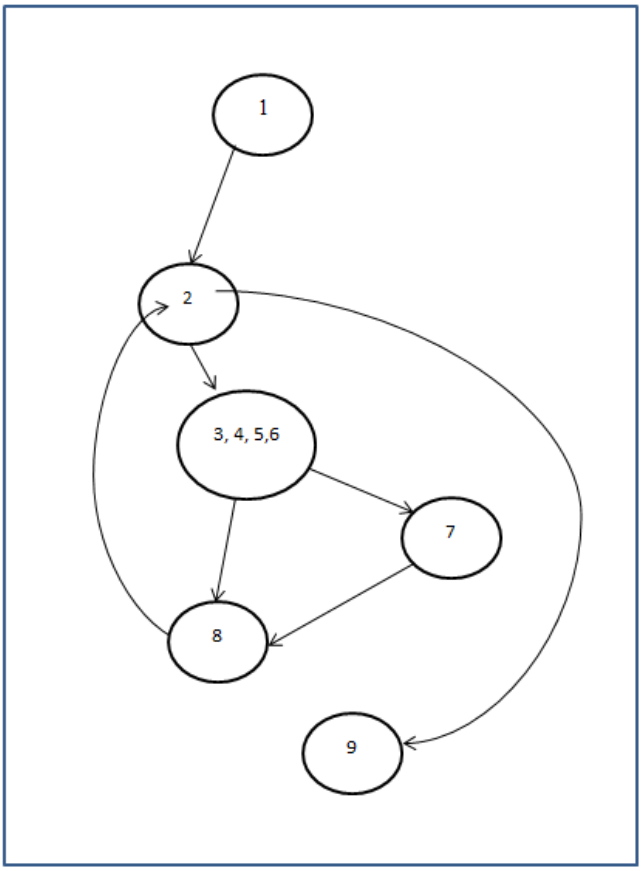


Figura 33: Diagrama Flujo de Control Función Abre pinza

Complejidad ciclomatica

G(s) = aristas – nodos + 2 = 7 – 6 + 2 = 3

La complejidad ciclomática del sistema es de 3 caminos de prueba.

Camino 1:	1,2,9
Camino 2:	1,2,3,4,5,6,7,8,2,9
Camino 3:	1,2,3,4,5,6,8,2,9

Fuente: Elaboración propia

Tabla 14: Caminos Función Abre pinza

II.17.2.4. Pruebas de camino básico: Función movimientos_Callback

```
function Movimentos_Callback(hObject, eventdata, handles)

global world Velocidad Rango_A Rango_B Rango_C Rango_G_P A B C P Edit_GradoA Edit_GradoB Edit_GradoC Edit_Giro_P AntiguoA AntiguoB AntiguoC AntiguoP

A=0;

B=0;

C=0;

P=0;

a0=AntiguoA;

b=AntiguoB;

c=AntiguoC;

p=AntiguoP;

for a=-pi:Velocidad:pi

%-------------------------Retorno de GradoA-----------------------------

if(AntiguoA~=Edit_GradoA)

if(Edit_GradoA<0)

%decremento de barra

if(AntiguoA>Edit_GradoA)

if(a<=AntiguoA && a>=Edit_GradoA)%Para todo Edit_GradoA<=a<=AntiguoA

Rango_A=Edit_GradoA;

Grado_A=vrnode(world,'Brazo_3Grados');%cargando nodo de VRML

Grado_A.rotation=[0 1 0 a0];

set(handles.GradoA,'Value',a0);
```

```
A=round(a0*180/pi);
set(handles.Grado_A,'String',A); %Escribe el valor de
a0=a0-Velocidad;
vrdrawnow;
drawnow
end
else
%incremento de barra
if(a>=AntiguoA && a<=Edit_GradoA)%Para todo AntiguoA<=a<=Edit_GradoA
a0=a;
Rango_A=Edit_GradoA;
Grado_A=vrnode(world,'Brazo_3Grados');%cargando nodo de VRML
Grado_A.rotation=[0 1 0 a0];
set(handles.GradoA,'Value',a0);
A= round(a0*180/pi);
set(handles.Grado_A,'String',A); %Escribe el valor de
vrdrawnow;
drawnow
end
end


%%retorno desede posición mmayores a 0 (0,180)
```

```
else
if(AntiguoA>Edit_GradoA)
%decremento de barra
if(a<=AntiguoA && a>=Edit_GradoA)%Para todo AntiguoA=>a>=Edit_GradoA
Rango_A=Edit_GradoA;
Grado_A=vrnode(world,'Brazo_3Grados');%cargando nodo de VRML
Grado_A.rotation=[0 1 0 a0];
set(handles.GradoA,'Value',a0);
A=round(a0*180/pi);
set(handles.Grado_A,'String',A); %Escribe el valor de
a0=a0-Velocidad;
vrdrawnow;
drawnow
end
else
%incremento de barra
if(a<=Edit_GradoA && a>=AntiguoA) %Para todo Edit_GradoA=>a>=AntiguoA
a0=a;
Rango_A=Edit_GradoA;
Grado_A=vrnode(world,'Brazo_3Grados');%cargando nodo de VRML
Grado_A.rotation=[0 1 0 a0];
set(handles.GradoA,'Value',a0);
A= round(a0*180/pi);
```

```
set(handles.Grado_A,'String',A); %Escribe el valor de
vrdrawnow;
drawnow
end
end
end
end

%-------------------------Retorno de GradoB-----------------------------
if(AntiguoB~=Edit_GradoB)
if(AntiguoB>Edit_GradoB)
%decremento de barra
if(a<=AntiguoB && a>=Edit_GradoB)
Rango_B=Edit_GradoB;
Grado_B=vrnode(world,'Grado2');%cargando nodo de VRML
Grado_B.rotation=[0 0 1 b];
set(handles.GradoB,'Value',b);
B=round(b*180/pi);
set(handles.Grado_B,'String',B); %Escribe el valor de
b=b-Velocidad;
vrdrawnow;
drawnow
end
```

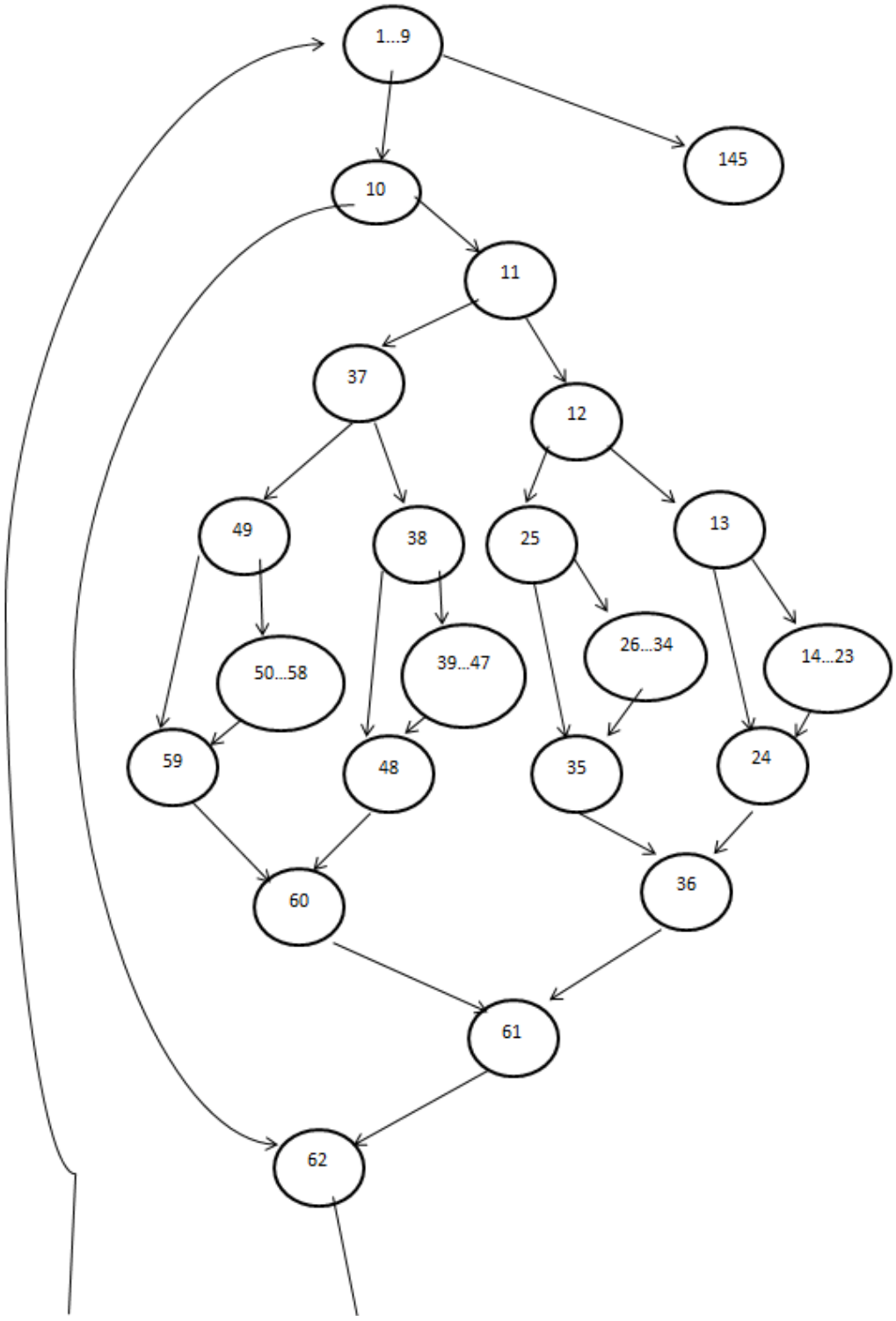
```
else
%incremento de barra
if(a<=Edit_GradoB && a>=AntiguoB)
Rango_B=Edit_GradoB;
b=a;
Grado_B=vrnode(world,'Grado2');%cargando nodo de VRML
Grado_B.rotation=[0 0 1 b];
set(handles.GradoB,'Value',b);
B=round(b*180/pi);
set(handles.Grado_B,'String',B); %Escribe el valor de
vrdrawnow;
drawnow
end
end
end
%-------------------------Retorno de GradoC------------------------------
if(AntiguoC~=Edit_GradoC)
if(AntiguoC>Edit_GradoC)
%decremento de barra
if(a<=AntiguoC && a>=Edit_GradoC)
Rango_C=Edit_GradoC;
Grado_C=vrnode(world,'Grado3');%cargando nodo de VRML
Grado_C.rotation=[0 0 1 c];
```

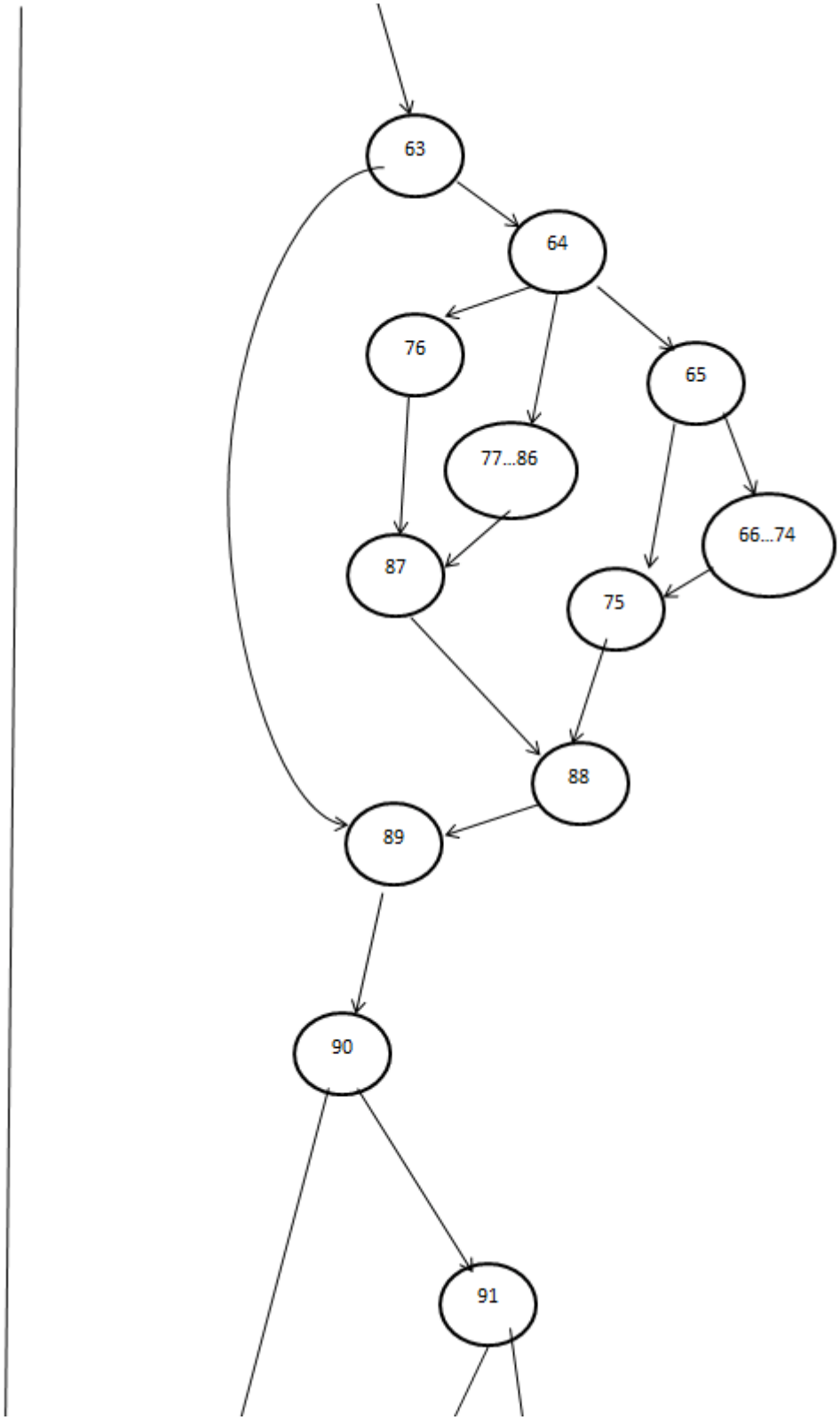
```
set(handles.GradoC,'Value',c);
C=round(c*180/pi);
set(handles.Grado_C,'String',C); %Escribe el valor de
c=c-Velocidad;
vrdrawnow;
drawnow
end
else
%incremento de barra
if(a<=Edit_GradoC && a>=AntiguoC)
Rango_C=Edit_GradoC;
c=a;
Grado_C=vrnode(world,'Grado3');%cargando nodo de VRML
Grado_C.rotation=[0 0 1 c];
set(handles.GradoC,'Value',c);
C=round(c*180/pi);
set(handles.Grado_C,'String',C); %Escribe el valor de
vrdrawnow;
drawnow
end
end
end
%-------------------------Retorno de Pinza-----------------------------
```

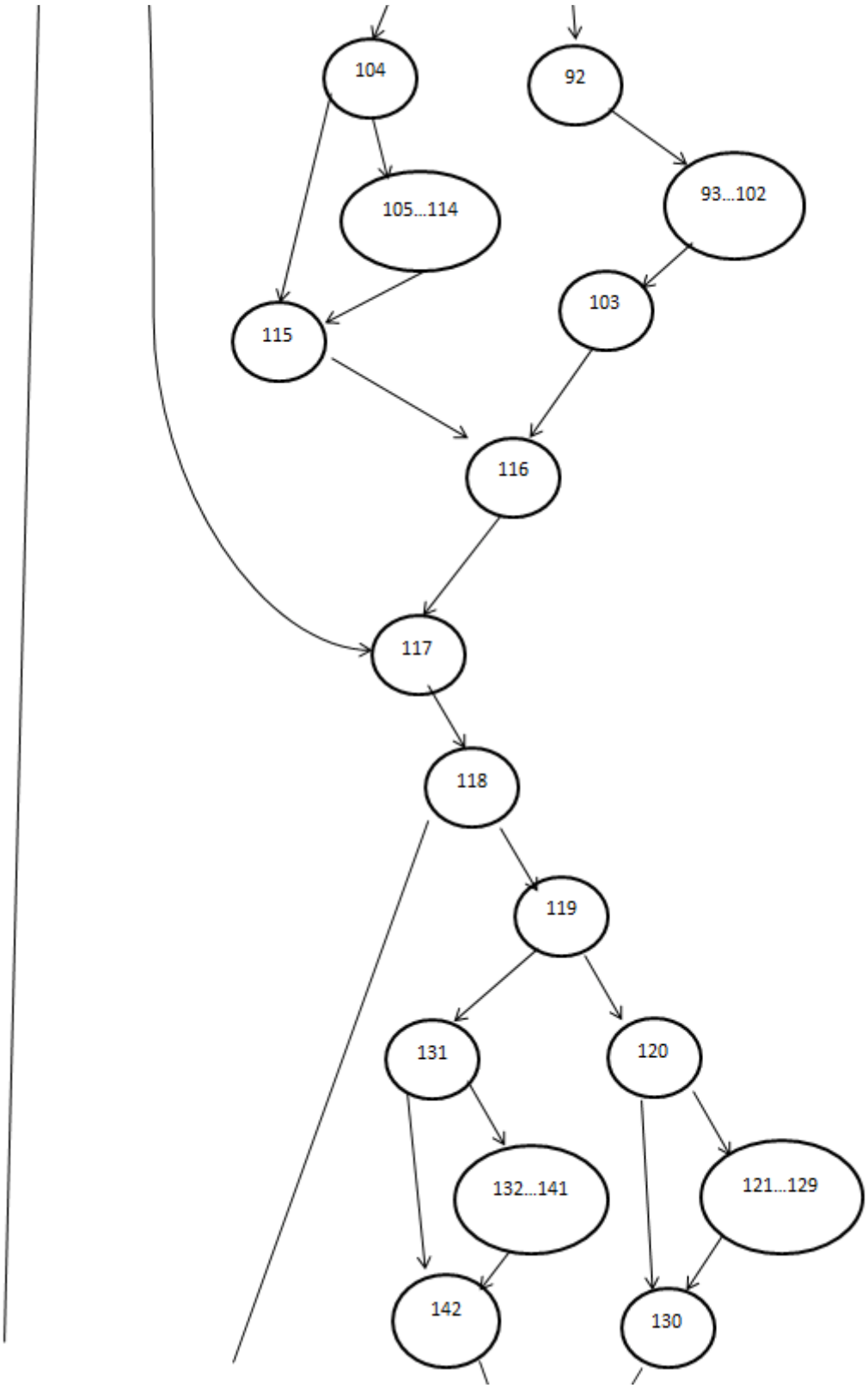
```
if(AntiguoP~=Edit_Giro_P)
if(AntiguoP>Edit_Giro_P)
%decremento de barra
if(a<=AntiguoP && a>=Edit_Giro_P)
Rango_G_P=Edit_Giro_P;
Grado_P=vrnode(world,'Pinza');%cargando nodo de VRML
Grado_P.rotation=[0 1 0 p];
set(handles.GIRO_PINZA,'Value',p);
P=round(p*180/pi);
set(handles.Giro_P,'String',P); %Escribe el valor de
p=p-Velocidad;
vrdrawnow;
drawnow
end
else
%incremento de barra
if(a<=Edit_Giro_P && a>=AntiguoP)
Rango_G_P=Edit_Giro_P;
p=a;
Grado_P=vrnode(world,'Pinza');%cargando nodo de VRML
Grado_P.rotation=[0 1 0 p];
set(handles.GIRO_PINZA,'Value',p);
P=round(p*180/pi);
```

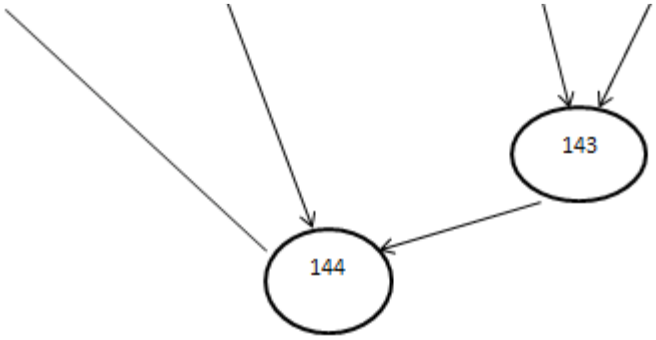
```
set(handles.Giro_P,'String',P); %Escribe el valor de

vrdrawnow;

drawnow

end

end

end


end

AntiguoA=Edit_GradoA;

AntiguoB=Edit_GradoB;

AntiguoC=Edit_GradoC;

AntiguoP=Edit_Giro_P;
```

Diagrama de Flujo de Control




63
64
76
65
77...86
66...74
87
75
88
89
90
91


104
92
105…114
93…102
103
115
116
117
118
119
131
120
132…141
121…129
142
130



Fuente: Elaboración propia

Figura 34: Diagrama de Flujo de Control Función Movimientos

Complejidad ciclomatica

G(s) = aristas – nodos + 2 = 72 – 52 + 2 = 22

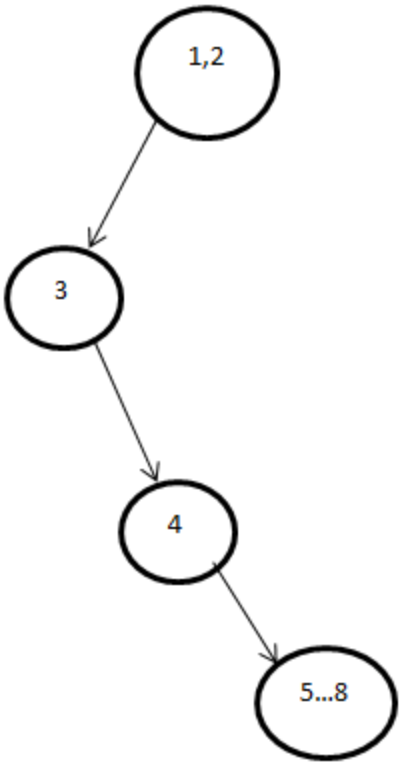
La complejidad ciclomática del sistema es de 22 caminos de prueba

II.17.2.5. Pruebas de camino básico : Función Edit_GradoA_Callback

```
function Edit_GradoA_Callback(hObject, eventdata, handles)
global Edit_GradoA Edit_GradoB Edit_GradoC Edit_Giro_P AntiguoB AntiguoC AntiguoP
Edit_GradoA=str2double(get(hObject,'String')); %Almacenar valor ingresado y Transformar a formato double
%identifica valor no numerico
if isnan(Edit_GradoA)
errordlg('El valor debe ser numérico','ERROR');
set(handles.Edit_GradoA,'String','');
return
end
%identificando rango
if(Edit_GradoA<-120 ||Edit_GradoA>170)
errordlg('El Rango  incorrecto (-120,170)','ERROR');
set(handles.Edit_GradoA,'String','');
return
end
Edit_GradoA=((Edit_GradoA*pi)/180);
%ASEGURANDO SOLO EL MOVIMIENTO DE BARAS A
Edit_GradoB=AntiguoB;
Edit_GradoC=AntiguoC;
Edit_Giro_P=AntiguoP;
```

Movimentos_Callback(hObject, eventdata, handles)

Diagrama de Flujo de Control



Fuente: Elaboración Propia

Figura 35: Diagrama de Flujo de Control Funcion Edit_GradoA_Callback

Complejidad ciclomática

G(s) = aristas – nodos + 2 = 3 – 4 + 2 = 1

La complejidad ciclomatica del sistema es de 1 caminos de prueba.

Camino 1:	1,2,3,4,5…8

Fuente: Elaboración propia

Tabla 15: Caminos Función Edit GradoA

II.17.2.6. Plan del Proyecto

II.17.2.6.1. Plan de las Fases

El desarrollo se llevará a cabo en base a fases con una o más iteraciones en cada una de ellas. La siguiente tabla muestra distribución de tiempo y el número de iteraciones de cada fase.

Fase	Nro. Iteraciones	Duración (días)
Fase de Inicio y elaboración	1	33
Fase de Construcción	1	114
Fase de Transición	1	20

Fuente: Elaboración propia

Tabla 16: Plan de Fases

Los hitos que marcan el final de cada fase se describen en la siguiente tabla.

Descripción	Hito
Fase de inicio y elaboración.	*Hito: Arquitectura del ciclo de vida.* En esta fase de desarrollo es de recopilación de la información, sobre los recursos con los que se cuenta (instalaciones, brazo robot, software.) e investigar características técnicas del brazo robótico, más los accesorios

	utilizados y modelado de los casos de uso. Además se analizan los requisitos y se desarrolla un protocolito de arquitectura (incluyendo las partes más relevantes y/o críticas del sistema.) al final de esta fase, todos los casos de uso correspondientes a requisitos que serán implementados en la reléase (liberación o entrega de software) de la fase de Construcción deben estar analizados y diseñados en el (Modelo análisis / Diseño). La revisión y aceptación del prototipo de la arquitectura del sistema marca el final de esta fase. La primera iteración tendrá como objetivo la identificación y especificación de los principales casos de uso, así como su realización preliminar en el Modelo de Análisis / Diseño, también permitirá hacer una revisión general del estado de los artefactos hasta este punto y ajustar si es necesario la planificación para asegurar el complimiento de los objetivos.
Fase de Construcción.	*Hito: Capacidad operacional.* Durante la fase se construye los componentes restantes y se incorporan a

	la investigación. Se realizan los modelos de implementación y despliegue, corriendo en plataforma. Se comienza la elaboración de material de apoyo al usuario. El hito que marca el fin de esta fase es la capacidad operacional parcial del producto que se haya considerado como crítica, lista para ser entregado a los usuarios para pruebas beta.
Fase de transición	*Hito: Producto* En esta fase se prepara el traspaso del software desarrollado a la comunidad de usuarios, incluyendo el entrenamiento de los usuarios. El hito que marca el fin de esta fase incluye, la entrega de toda la documentación del proyecto con el material de apoyo al usuario, la finalización del entrenamiento de los usuarios.

Fuente: Elaboración propia

Tabla 17: Fases de diseño.

II.17.2.7. Estimación del proyecto.

Se desea estimar los costos que tendrá el sistema "Control del Brazo Robótico Scorbot er4u mediante Matlab"

Se utilizará el método COCOMO para estimación de costos y para esto se tiene los siguientes datos: el sistema estima producir 0.7

KLDC se pretende que el sistema tenga una complejidad media donde se pretenderá cobrar 200 $us.

Método Cocomo Básico

Desarrollo	
Datos tamaño	**0.7 KLDC**
Costo	**EN* tarifa mensual**
Para calcular el esfuerzo nominal se va a utilizar la siguiente fórmula.	
EN	**$2.4*KLDC^{1.05}$**
	$2.4*(0.7)^{1.05}=1.65$ **= 1.65 personas / mes**
Tiempo de desarrollo TDES =	**$2.5\ (EN)^{0.38} = 2.5\ (1.65)^{0.38}$** **= 3.024 = 3 meses.**
Calcular número de personas	
N	**EN/TDES = 1.65 / 3 = 0.55** **= 1**

Fuente: Elaboración propia

Tabla 18: Estimación del proyecto

Detalle	Unidad	Cantidad	Mensual	Total
Jefe del	1	3 meses	250	750

proyecto				
Ingeniero de Software	1	3 meses	200	600
Analista de Sistemas	1	3 meses	200	600
Programador	1	3 meses	200	600
Luz e internet	1	3 meses	30	90
Total en Dólares				2640

Fuente: Elaboración propia

Tabla 19: Costo de Desarrollo

II.18. Capacitación en el uso del sistema "manejo del brazo robot scorbot er4u mediante matlab"

II.18.1. Introducción

El objetivo de este componente es capacitar a los que operarán el brazo robot del "manejo del brazo robot scorbot er4u mediante matlab" según el nivel de los mismos empleando métodos y medios de enseñanza-aprendizaje adecuados.

El propósito del proyecto es "Control del brazo robótico SCORBOT er 4u. Mediante matlab", la capacitación en el uso del sistema informático al personal afectado por el proyecto se convierte en un componente fundamental para el logro del mismo.

El componente capacitación, se encamina hacia el siguiente objetivo: usar adecuadamente el sistema informático desarrollado, mejorando la capacidad de entendimiento del operador en su funcionamiento del mismo.

La capacitación será presencial dada la corta duración del mismo, la disponibilidad de ambientes, de materiales didácticos y la importancia del posibilitar que el operador reciba asesoramiento oportuno ante cualquier consulta.

Además la capacitación se refiere a las metodologías que se usan para proporcionar a las personas que estarán involucradas con este proyecto (Universitario, docente), las habilidades necesarias que necesitan para realizar su trabajo de una manera más eficiente, esto contempla desde un pequeño curso que le permitirán al usuario entender el funcionamiento básico del sistema nuevo, y una capacitación más profunda y avanzada a base de prácticas y material didáctico.

II.18.2. Definición de capacitación

La capacitación es un proceso educacional de carácter estratégico aplicado de manera organizada y sistémica, mediante el cual los colaboradores adquieren o desarrollan conocimientos y habilidades especificas relacionadas al trabajo, y modifica sus actividades frente a los quehaceres de la organización, el puesto o el ambiente laboral.

II.18.3. Contexto

La capacitación se desarrollará en un curso de una hora de duración, 1 sola vez en una semana determinada, se dividieron los puntos de estudio de tal forma que se afronta el tema de una forma global, y llegando a lo específico el control de brazo robot.

II.18.4. Propuesta Pedagógica

La propuesta pedagógica a utilizar dada las características de los usuarios del sistema informático desarrollado, tendrá en cuenta sus conocimientos previos y como estos podrán ayudarlos a desarrollar otros posibles proyectos.

Los métodos de enseñanza a utilizar pondrán su énfasis principalmente en dos teorías de aprendizaje: la cognitiva, con su máximo exponente en el constructivismo, la colaborativa, fundamentalmente para ser explotada con intensidad en la formación del personal técnico.

Finalmente se pone de manifiesto el aprendizaje significativo porque el operador tiene que incorporar los nuevos conocimientos en forma sustantiva en su estructura cognitiva. Esto se logra cuando el usuario relaciona los nuevos conocimientos con los anteriormente adquiridos, pero también es necesario que el usuario se interese por aprender lo que se le está mostrando. De esta forma el usuario no solo obtendrá resultados satisfactorios en un trabajo final, sino que será capaz de enfrentarse a diversas situaciones y aplicar los conocimientos adquiridos.

II.18.5. Metodología

La metodología utilizada para esta capacitación i/o taller se basa en el modelo pedagógico “Pedagogía no directiva”, donde el enseñar es propiciar las condiciones para que el estudiante exprese libremente sus necesidades en un clima afectivo favorable de comprensión, aceptación y respeto, y el aprender es atribuirle significación a la experiencia que posibilita la satisfacción de las necesidades..

Es una metodología, que busca desarrollar su sensibilidad frente a problemas reales, estudiar alternativas de solución y evaluar sus implicaciones en conjunto con la utilización de tecnologías.

II.18.6. Contenidos de la capacitación

Tema 1.- Introducción a la robótica industrial, diseño y control de una célula robotizada, seguridad en instalaciones robóticas, Operaciones realizadas con el software desarrollado y el brazo robot scorbot er 4u.

II.18.7. Plan de Clases

NRO.	CONTENIDO	OBJETIVO	FECHA	DURACION (HORAS)	MATERIAL DIDACTICO	MEDIOS DE ENSEÑANZA APRENDIZAJE	DESTINATARIO
1	Tema 1: Introducción a la robótica industrial. Funciones y operaciones realizadas con el software desarrollado.	• Que el operador tenga entendimiento de conceptos generales. ▪ Se mostrará en acción el uso del brazo robot con el sistema desarrollado.	02/09/13 05/09/13	1	Diapositivas y videos ilustrativos. Brazo robot scorbot er4u.	Disertación. Demostración.	Operadores.

Fuente: Elaboración propia

Tabla 20: Plan de Clases

II.18.8. Cronograma

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predece
1		**Recopilacion de la información y determinacion de requerimientos del operador**	33 días	mar 05/03/13	jue 18/04/13	
2		**Analisis y diseño del sistema**	**61 días**	**lun 18/03/13**	**dom 09/06/13**	**1**
3		SML (Ingenieria de requerimietos del sistema)	30 días	lun 18/03/13	vie 26/04/13	
4		Contrucción de los diagramas UML del sistema	25 días	lun 29/04/13	vie 31/05/13	3
5		Diseño y elaboración de interfaces de usuario	2 días	lun 03/06/13	mar 04/06/13	4
6		**Desarrollo del sistema**	**74 días**	**mié 17/04/13**	**sáb 27/07/13**	
7		Programación	60 días	mié 17/04/13	mar 09/07/13	
8		Casos de prueba del sistema (caja negra, caja blanca)	13 días	mié 10/07/13	sáb 27/07/13	7
9		**Se elabora el proyecto con el material de apoyo**	**31 días**	**sáb 27/07/13**	**lun 09/09/13**	
10		Elaboración de material y guías del curso de capacitación	20 días	sáb 27/07/13	jue 22/08/13	
11		Preparación de Ambientes de Capacitación	1 día	vie 23/08/13	vie 23/08/13	10
12		Realizar capacitación según autorización de la UAJMS	1 día	jue 05/09/13	jue 05/09/13	11
13		Realizar entrega de certificados de capacitación	1 día	vie 06/09/13	vie 06/09/13	12

Fuente: Elaboración propia

Tabla 21: Cronograma de trabajo

Fuente: Elaboración propia

Figura 36: Cronograma de trabajo 2

II.18.9. Resultados esperados

Al final de la ejecución del proyecto, en octubre del 2013 el 80% de los universitarios involucrados de 4to y 5to año se entran capacitados en el manejo del sistema.

II.18.10. Medios de Verificación del Componente

- Registro de participantes del curso taller, firmada por los participantes.
- Certificado de participación.

II.18.11. Conclusiones

Los operadores entienden el funcionamiento del sistema después de la capacitación.

III. Conclusiones y recomendaciones

III.1. Conclusiones

Con el uso de la MML y en base al trabajo realizado y la experiencia adquirida a medida que se desarrolló este proyecto se concluye que el Marco Lógico fue una buena forma de abordar el proyecto, porque se pudo presentar sistemática y lógica los objetivos del proyecto y sus relaciones de causalidad además de mejorar la planificación, el seguimiento y la evaluación del proyecto.

Habiendo desarrollado de manera adecuada las actividades previas se llegó a cumplir con el indicador de propósito, esto es importante, porque así evidenciamos que el sistema contribuye significativamente en el fin del proyecto "**Eficiencia en la ejecución de trabajos informático-robóticos en industrias, instituciones privadas y públicas**".

Con los resultados obtenidos a lo largo del proyecto demostramos que el uso de la metodología RUP fue la más adecuada, porque a través de la notación UML pudimos visualizar, especificar, construir y documentar el sistema de software desarrollado, además de aplicar situaciones de tiempo real con la notación.

La interfaz del usuario se desarrolló de manera intuitiva, empleando la información recopilada y mostrándolas en las pantallas con el propósito de que el operador pueda interactuar con el sistema sin dificultad. Por otra parte también fueron considerados los mecanismos de validación para evitar que se ingresen datos erróneos, garantizando una mayor confiabilidad de la información almacenada.

III.2. Recomendaciones

Recomiendo el uso de la metodología MML para este y próximos proyectos. Porque nos facilita el mecanismo apropiado para comprender lo que quiere el usuario o cliente en otros casos, y plasmarlos en el proyecto describiendo los servicios y las restricciones asociadas a él, ya que la documentación del proyecto fue desarrollada a partir del análisis de requisitos obtenidos a lo largo del proyecto y la MML.

Se recomienda que el equipo computacional del operador del sistema tenga instalado plataforma Windows XP o superior como mínimo.

Es aconsejable consultar el manual de usuario para cualquier duda en el uso del sistema y requerimiento técnicos de hardware y software del sistema para un adecuado uso.

Se deja como una recomendación la posibilidad de una conexión a una base de datos. Se propone como base de datos a MySQL con MatLab[1], que por cuestiones de tiempo no se pudo implementar en este proyecto y al pie de página se deja un ejemplo para realizar las acciones necesarias para su implementación.

[1] Zona Vertigo2040http://vertigo2040.wordpress.com/2011/01/13/conexion-de-mysql-en-matlab/