PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Reconocimiento de Posicionamiento Y Direccionamiento De lego robot a través de OpenCV.

Francisco Javier Ahumada Cortes.

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO CIVIL EN INFORMÁTICA

Carrera: **Ingeniería Civil Informática**

DICIEMBRE 2015

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Reconocimiento de Posicionamiento Y Direccionamiento De lego robot a través de OpenCV.

Francisco Javier Ahumada Cortes.

Profesor Guía: **Claudio Cubillos Figueroa**

Profesor Co-Referente: **Ismael Figueroa Palet**

Carrera: **Ingeniería Civil Informática**

DICIEMBRE 2015

Dedicado a mi madre Margarita Cortes Báez, mi padre José Ahumada Fernández y Hermano José Ahumada Cortes, por todo el apoyo brindado a lo largo de mis años de estudio y desarrollo del presente proyecto.

También a los profesores Claudio Cubillos Figueroa y Ismael Figueroa Palet, por la guía dada para poder desarrollar el presente proyecto.

Como último a los amigos que tendieron una mano en esos momentos necesarios para poder completar algunos problemas del proyecto.

Índice

Glosario de Términos ..... **iv**
Lista de abreviaturas o Siglas ..... **vi**
Lista de figuras ..... **vii**
Lista de tablas ..... **ix**
Resumen ..... **x**
Abstract ..... **x**
1 Descripción del proyecto **1**
1.1 Introducción **1**
1.2 Definición de objetivos **2**
1.2.1 Objetivo General **2**
1.2.2 Objetivos Específicos **2**
1.3 Plan de trabajo **2**
1.4 Metodología de trabajo **5**
1.4.1 Desarrollo Incremental **5**
2 Marco Teórico **7**
2.1 Visión Humana **7**
2.2 Visión artificial **7**
2.2.1 Fases de un sistema de visión artificial **8**
2.2.2 Componentes de un sistema de visión artificial **9**
2.3 Digitalización de Imagen **10**
2.3.1 Muestreo **10**
2.3.2 Cuantización **11**
2.4 Procesamiento de Imagen **12**
2.4.1 Aumento y reducción de contrastes **12**
2.4.2 Filtrado espacial **15**
2.4.3 Filtro de suavizado **16**
2.4.4 Filtros de obtención de contornos **17**
2.5 Open CV **18**
2.6 Agentes Inteligentes y Sistema Multi-agente. **19**
2.6.1 Agente **19**
2.6.2 Agente Inteligente. **19**

2.6.3 Sistema Multi-agente 20
2.6.4 Infraestructura para agentes 21
2.6.5 Comunicación entre agentes 21
2.6.6 Metodología PASSI 22
2.6.7 Plataforma JADE 24
2.7 Robótica 24
2.8 Lego Mindstorms 26
2.8.1 Ladrillo NXT 26
2.8.2 Microcontrolador 27
2.8.3 Servomotores 27
2.8.4 Sensores 28
2.9 LeJOS NXJ 28
3 Desarrollo del proyecto 30
3.1 Finalidad del proyecto 30
3.2 Desarrollo del Modelado del Sistema Multi-Agente. 31
3.2.1 Modelo de Requerimientos del sistema. 31
3.2.2 Modelo de sociedad de agentes 35
3.2.3 Modelo Implementación de Agentes 38
3.2.4 Modelo de Despliegue 39
3.3 Desarrollo del Sistema de reconocimiento de posicionamiento y direccionamiento de lego robot a través de OpenCV 40
3.3.1 Procedimiento previo a la implementación 40
3.3.2 Implementación 43
3.4 Casos de estudio y pruebas 46
3.4.1 Entorno de Prueba 46
3.4.2 Caso de estudio giro del NXT 47
3.4.3 Caso de estudio de Posicionamiento 48
3.4.4 Caso de estudio tomar balón 49
3.4.5 Caso de estudio pruebas de latencia 49
4 Conclusión 50
5 Referencias 51
Anexos 53
A. Desarrollo del Sistema de detección de triángulos 53
A.1 Implementación 53

A.2 Pruebas **56**
A.2.1 Prueba de detección de figura de una imagen **56**
A.2.2 Prueba de detección de figura por cámara **56**
A.2.3 Pruebas de obtención de información **57**
B. Desarrollo del Sistema Multi-Agente Robot-OpenCV **58**
B.1 Implementación **58**
B.2 Pruebas **62**
B.2.1 Entorno de Prueba **62**
B.2.2 Prueba de comunicación entre agentes **63**
B.2.3 Prueba de Posicionamiento en el área de prueba **63**
B.2.4 Pruebas de Giro del NXT **64**
C. Desarrollo Sistema multi-agente para interacción de dos Legos NXT. **65**
C.1 Implementación **65**
C.2 Pruebas **68**
C.2.1 Entorno de Prueba **68**
C.2.2 Prueba de toma de balón **70**
C.2.3 Prueba de Posicionamiento del balón **70**
C.2.4 Pruebas Sistema completo **70**
D. Pruebas de latencia **71**
D.1 Tabla prueba de latencia uno **71**
D.2 Tabla prueba de latencia dos **72**
D.3 Tabla prueba de latencia tres **74**

Glosario de Términos

Agente: un agente se entiende como Toda entidad que de forma autónoma perciba su entorno (real o simulado) mediante sensores y que actúe en el mismo mediante efectores.

Agentes Inteligentes: un agente inteligente se define como todo ente que es capaz de realizar acciones autónomas y flexibles en la búsqueda de objetivos para los cuales fue diseñado.

Bluetooth: es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz.

Contraste se define como la diferencia relativa en la intensidad entre un punto de una imagen. Un ejemplo simple es el contraste entre un objeto de brillo constante sobre un fondo de un brillo constante.

Driver: llamado normalmente controlador es un programa informático que permite al sistema operativo interactuar con un periférico, haciendo una abstracción del hardware y proporcionando una interfaz para usarlo.

Firmware: El firmware es un bloque de instrucciones de máquina para propósitos específicos, grabado en una memoria de tipo de solo lectura, por ejemplo ROM, que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo.

Inteligencia Artificial: En ciencias de la computación se denomina inteligencia artificial a la capacidad de razonar de un agente no vivo.

JAVA: Lenguaje de programación orientado a objetos. Con él se pueden desarrollar aplicaciones para distintos tipo de dispositivos y plataformas.

Lego Mindstorms NXT: kit de robótica desarrollado por la empresa Lego los cuales pueden ejecutar tareas a través de sus actuadores y percibir el ambiente a través de sus sensores.

LeJOS NXJ: Firmware y conjunto de utilidades y clases Java que conforman un entorno de programación Java completo, para escribir programas orientado a objetos para Lego Mindstorms NXT.

Ontología: vocabulario acerca de un dominio: términos, relaciones, reglas de combinación para extender el vocabulario.

Plugin: complemento; es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API.

Protocolos: En informática, un protocolo es un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red por medio de intercambio de mensajes. Éste es una regla o estándar que controla o permite la comunicación en su forma más simple,

puede ser definido como las reglas que dominan la sintaxis, semántica y sincronización de la comunicación.

Robot: su origen viene de la palabra checa robota. Máquina o ingenio electrónico programable, capaz de manipular objetos y realizar operaciones antes reservadas solo a las personas.

Segmentación: En el campo de la visión artificial es el proceso de dividir una imagen digital en varias partes (grupos de píxeles) u objetos. El objetivo de la segmentación es simplificar y/o cambiar la representación de una imagen en otra más significativa y más fácil de analizar.

Sistemas Multi-Agente: se puede entender como una red de problem-solvers que trabajan conjuntamente para encontrar respuestas a problemas que ven más allá de las capacidades o conocimientos individuales de cada entidad.

Lista de abreviaturas o Siglas

SVA: Sistema de visión artificial.

DPI: pixeles por pulgada.

OpenCV: Open Source Computer Vision.

BSD: Berkeley Software Distribution.

SMA: sistema multi-agente.

ACL: Agent Communication Language.

RAE: Real Academia de la lengua Española.

KQML: Knowledge Query and Manipulation.

FIPA-ACL: Foundation for Intelligent Physical Agents - Agent Communication Language.

KIF: Knowledge Interchange Format.

FIPA-SL: FIPA Semantic Language.

PASSI: Process for Agent Societies Specification and Implementation.

UML: Unified Modeling Language.

OCL: Object Constraint Language.

Hw: hardware.

JADE: Java DEvelopment Framework.

FIPA: Foundation for Intelligent Physical Agents.

GUI: Graphical user interface.

RAM: Random Access Memory.

GPS: Global Positioning System.

JDK: Java Development Kit.

JavaCV: Java Computer Vision.

USB: universal serial bus.

Lista de figuras

Figura 1.1 Modelo de ciclo de vida incremental [1] 5
Figura 2.1 Diagrama de bloques de las etapas típicas en un sistema de visión artificial [5]. 9
Figura 2.2 Diagrama de bloques de un sistema SVA [4]. 10
Figura 2.3 Diferentes tipos de imágenes digitales y su tamaño en bytes [5]. 12
Figura 2.4 Histograma con poco contraste (izquierda), imagen saturara (derecha) [5]... 13
Figura 2.5 De izquierda a derecha la función lineal, cuadrada y raíz cuadrada [5]. 13
Figura 2.6 Funciones de transferencia para aumento y disminución de contraste [5]. 14
Figura 2.7 Transformaciones del histograma: (a) imagen original con su correspondiente histograma; (b) resultado de una operación de disminución de contraste; (c) aumento de contraste [5]. 15
Figura 2.8 Aplicación de un filtrado espacial de suavizado [5]. 17
Figura 2.9 Filtrado de Sobel en la dirección x en valor absoluto [5]. 18
Figura 2.10 Agente y su Ambiente [elab. Propia] 19
Figura 2.11 Esquema de modelos de fases PASSI [19] 24
Figura 2.12 Wabot-1[20] 25
Figura 2.13 Sojourner[20] 26
Figura 2.14 Ladrillo Lego Mindstorms NXT [22] 27
Figura 2.15 Servomotores [22] 27
Figura 2.16 Sensores[elab. Propia] 28
Figura 3.1 esquema escenario Final 31
Figura 3.2 Descripción de dominio 32
Figura 3.3 Identificación de Agentes 33
Figura 3.4 Identificación de Roles 34
Figura 3.5 Especificación de Tareas 35
Figura 3.6 Descripción de Ontología del Dominio 35
Figura 3.7 Descripción de ontología de Comunicación. 36
Figura 3.8 Descripción de Roles. 37
Figura 3.9 Descripción de Protocolos 37
Figura 3.10 Definición de la estructura Multi-Agente 38
Figura 3.11 Definición de la estructura del Agente Individual, AgenteCV 39
Figura 3.12 Modelo de Despliegue 39
Figura 3.13 Instalación de LeJOS 41

Figura 3.14 importación librerías **42**
Figura 3.15 Resultado importación librerías **42**
Figura 3.16 Interfaz del Sistema **43**
Figura 3.17 Entorno de Prueba **46**
Figura 3.18 Lego Mindstorms NXT de prueba y Balón **47**
Figura 3.19 Diagrama camino en L **48**
Figura 3.20 Diagrama camino directo **48**
Figura A.1 Captura de pantalla **54**
Figura A.2 Imagen al aplicar los filtros **54**
Figura A.3 Imagen detección de base **55**
Figura A.4 Imagen resultado final **56**
Figura B.1 Cancha Virtual del Sistema **61**
Figura B.2 Sistema Funcionando **61**
Figura B.3 Entorno de Prueba **62**
Figura B.4 Lego Mindstorms NXT de prueba **63**
Figura C.1 Lego NXT Proyecto costado **65**
Figura C.2 Lego NXT Proyecto frente **65**
Figura C.3 Dirección de NXTs **68**
Figura C.4 Entorno de Prueba **69**
Figura C.5 Lego Mindstorms NXT de prueba y Balón infrarrojo **69**

Lista de tablas

Tabla 1.1 Carta Gantt del proyecto parte uno 3
Tabla 1.2 Carta Gantt del proyecto parte dos 4
Tabla D.1 Prueba de latencia uno 71
Tabla D.2 Prueba de latencia dos 72
Tabla D.3 Prueba de latencia tres 74

Resumen

Para Aristóteles, la visión consistía en "*saber que hay y donde mediante la vista*". Con la vista el ser humano puede realizar tareas de la vida diaria que sin ellas serían más complejas, como lo es el poder caminar y saber dónde se va a dar el siguiente paso, el no chocar con los objetos, entre otras actividades. La visión artificial intenta simular este sentido, para así poder detectar diferentes objetos en el medio ambiente por medio de una cámara. Esto junto a la robótica es una combinación que puede simplificar mucho la vida de los seres humanos.

En este proyecto se desarrollará un sistema multi-agente el cual permita crear una visión artificial para los Lego Mindstorms NXT, para poder ser integrado a otros sistemas los cuales ocupen este tipo de robot, así generando una mejor precisión en los movimientos que realicen y así poder cumplir con sus tareas de una forma óptima.

Palabras Claves: Visión artificial, procesamiento de imagen, Lego Mindstorms NXT, sistema multi-agente.

Abstract

For Aristotle, the vision was "to know that there are and where by sight". With the sight the human being can perform daily tasks that without them would be more complex, as being able to walk and know where to make the next step, not hitting the objects, among other activities. Machine vision attempts to simulate this regard, in order to detect different objects in the environment by means of a camera. This, together with robotics is a combination that can greatly simplify the lives of human beings.

In this project, a multi-agent system which allows creating an artificial vision for Lego Mindstorms NXT developed. To be integrated with other systems which use this type of robot, generating a better accuracy in the movements they make and thus fulfill their tasks in a better way.

Keywords: computer vision, image processing, Lego Mindstorms NXT, multi-agent system.

1 Descripción del proyecto

1.1 Introducción

Desde el origen de la robótica el ser humano ha visto a ésta como un recurso con el que puede mejorar su forma de vida y adquirir mayor conocimiento. Esto debido a que con ésta se pueden desarrollar tareas rutinarias que gastan horas de producción en una empresa, que son peligrosas para el ser humano. También es posible simular diferentes ambientes para estudios científicos como en la medicina, para recrear enfermedades y estudiar sus tratamientos. En misiones de exploraciones marinas o espaciales para reconocer terrenos que pondrían en peligro al ser humano.

En este proyecto se enfocará mayormente en la visión artificial. Existen diferentes tipos de definición para visión, pero con todas estas se puede llegar a la conclusión que este sentido es uno de los más importante para el ser humano, esto debido a que la visión es la que permite saber qué hay y dónde está cada cosa que nos rodea. Esto se logra a través del procesamiento de imagen, que se produce al momento de captar una imagen con la vista y su posterior llegada al celebro.

La visión artificial intenta poder simular esta misma, para así poder mejorar los resultados en la práctica de robótica móvil, vehículos autónomos, manufactura, análisis e interpretación de imágenes, interpretación de escrituras, análisis de imágenes microscópicas, entre otras aplicaciones. Para intentar esta semejanza con la visión humana se han producido diversos estudios para la creación de filtros, con el fin de poder suavizar las imágenes, eliminar ruido, entre otros. El fin de esto es poder obtener a través de una cámara de computadora una imagen lo más similar a la que el ojo humano obtiene.

En la actual entrega se tiene como finalidad poder crear un sistema multi-agente, el cual se integrada con el sistema entregado en Proyecto 1. Con esto se pretende poder crear un sistema en cual un Lego NXT pueda posicionarse en un escenario, lo más exactamente a la posición que tiene que estar.

También se desarrollará el marco teórico, para poder entender de una mejor manera todo el desarrollo que se utilizó en el proyecto, como lo es el procesamiento y captura de imagen, cómo funcionan los agentes, funcionamiento de Lego mindstorms, entre otros. Para finalizar se explicada paso a paso todo el desarrollo que se realizó para desarrollar este sistema, junto con las pruebas necesarias, para poder realizar un proyecto sin ninguna falla al momento de hacerlo funcionar.

1.2 Definición de objetivos

En esta sección se nombrará el objetivo general del proyecto y todos los objetivos específicos que se tienen que llevar a cabo para cumplir la finalidad del proyecto.

1.2.1 Objetivo General

- Desarrollar un sistema multi-agente que permita la detección del direccionamiento y posicionamiento de robots Lego Mindstorms NXT dentro de un escenario.

1.2.2 Objetivos Específicos

.

- Estudiar el procesamiento de imágenes a través de OpenCV y el procesamiento de imágenes enfocado en la localización y dirección de contornos.
- Estudiar posibles problemas que puedan surgir, para diseñar casos de pruebas.
- Desarrollar escenarios de pruebas con Legos NXT.
- Desarrollar el sistema de direccionamiento y posicionamiento para los NXT.
- Validar el sistema final, por medio de los escenarios de prueba.

1.3 Plan de trabajo

Uno de los pasos más importantes al momento de desarrollar cualquier tipo de proyecto es la implementación de un plan de trabajo, esto debido a que gracias a esto se puede tener un orden y saber los tiempos que quedan para terminar el proyecto. A continuación se muestra la carta Gantt del presente proyecto, donde las tareas más importantes son el desarrollo de los sistemas. En las siguientes cartas Gantt no se tomaron en cuenta las fechas de presentaciones, debido al no conocimiento exacto de estas.

Tabla 1.1 Carta Gantt del proyecto parte uno

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		Estudio de proyectos anteriores	5 días	lun 16-03-15	vie 20-03-15	
2		Investigación sobre OpenCV	5 días	lun 23-03-15	vie 27-03-15	1
3		investigación sobre Visión artificial	5 días	lun 23-03-15	vie 27-03-15	1
4		Investigación sobre Digitalización de imagen	5 días	lun 23-03-15	vie 27-03-15	1
5		Investigación sobre procesamiento de imagen	5 días	lun 23-03-15	vie 27-03-15	1
6		Investigación sobre multi-agente	5 días	lun 23-03-15	vie 27-03-15	1
7		Redaccion informe de avance proyecto1	19 días	lun 30-03-15	jue 23-04-15	6
8		Desarrollo Sistema deteccion triángulos	15 días	lun 30-03-15	vie 17-04-15	
9		Pruebas del sistema	14 días	mié 01-04-15	lun 20-04-15	
10		Presentacion Avance Proyecto 1	1 día	mié 22-04-15	mié 22-04-15	
11		Desarrollo diagramas multi-agente	5 días	mié 22-04-15	mar 28-04-15	
12		Arreglos de avance proyecto 1	3 días	lun 27-04-15	mié 29-04-15	
13		Investigació LeJos NXJ	5 días	jue 30-04-15	mié 06-05-15	12
14		Investigación sobre JADE	5 días	jue 07-05-15	mié 13-05-15	13
15		Desarrollo Sistema deteccion NXT	27 días	jue 14-05-15	vie 19-06-15	14
16		Pruebas del sistema	22 días	jue 21-05-15	vie 19-06-15	
17		Redaccion informe final proyecto1	27 días	jue 14-05-15	vie 19-06-15	14
18		Presentacion Final Proyecto 1	1 día	lun 22-06-15	lun 22-06-15	
19		Estudio de código fuente de proyectos anteriores	5 días	mar 01-09-15	lun 07-09-15	
20		Estudio respecto a manejo de balón infrarrojo	7 días	mar 08-09-15	mié 16-09-15	19
21		Investigación de modelos de jugadores	10 días	jue 17-09-15	mié 30-09-15	20

Tabla 1.2 Carta Gantt del proyecto parte dos

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
22		Construcción Robot futbolistas	5 días	jue 01-10-15	mié 07-10-15	21
23		Construcción de sistema detección de balón infrarrojo	3 días	vie 02-10-15	mar 06-10-15	
24		Pruebas sistema detección de balón infrarrojo	2 días	jue 08-10-15	dom 11-10-15	23
25		Construcción de sistema posicionamiento de balón	2 días	lun 12-10-15	mar 13-10-15	24
26		Pruebas sistema posicionamiento de balón	2 días	mié 14-10-15	jue 15-10-15	25
27		Integración sistema completo	2 días	vie 16-10-15	lun 19-10-15	26
28		Pruebas sistema completo	1 día	mar 20-10-15	mar 20-10-15	27
29		Redacción informe avance	2 días	mié 21-10-15	jue 22-10-15	28
30		Presentacion avance proyecto 2	1 día	vie 23-10-15	vie 23-10-15	
31		Implementación sistema giro	3 días	lun 26-10-15	mié 28-10-15	
32		Pruebas sistema giro	2 días	jue 29-10-15	dom 01-11-15	31
33		Implementación sistema posicionamiento	2 días	lun 02-11-15	mar 03-11-15	
34		Pruebas Sistema Posicionamiento	2 días	mié 04-11-15	jue 05-11-15	
35		Desarrollo modelos Multi-agente sistema final	6 días	jue 05-11-15	jue 12-11-15	
36		Implementación sistema deteccion de circunferencia	2 días	jue 12-11-15	vie 13-11-15	
37		Pruebas del sistema deteccion de circunferencia	2 días	sáb 14-11-15	lun 16-11-15	36
38		Implementación Sistema final	10 días	mar 17-11-15	lun 30-11-15	37
39		Pruebas Sistema Final	3 días	mar 01-12-15	jue 03-12-15	38
40		Demostración Sistema Final	1 día	vie 04-12-15	vie 04-12-15	39
41		Corrección errores sistema final	5 días	lun 07-12-15	dom 13-12-15	40
42		Redacción sistema Final	5 días	lun 14-12-15	vie 18-12-15	
43		Presentacion Final proyecto 2	1 día	lun 21-12-15	lun 21-12-15	42

1.4 Metodología de trabajo

1.4.1 Desarrollo Incremental

El modelo incremental combina elementos del modelo en cascada con la filosofía interactiva de construcción de prototipos. Se basa en la filosofía de construir incrementando las funcionalidades del programa. Este modelo aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Cada secuencia lineal produce un incremento del software.

Cuando se utiliza un modelo incremental, el primer incremento es a menudo un producto esencial, sólo con los requisitos básicos. Este modelo se centra en la entrega de un producto operativo con cada incremento. Los primeros incrementos son versiones incompletas del producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación [1].

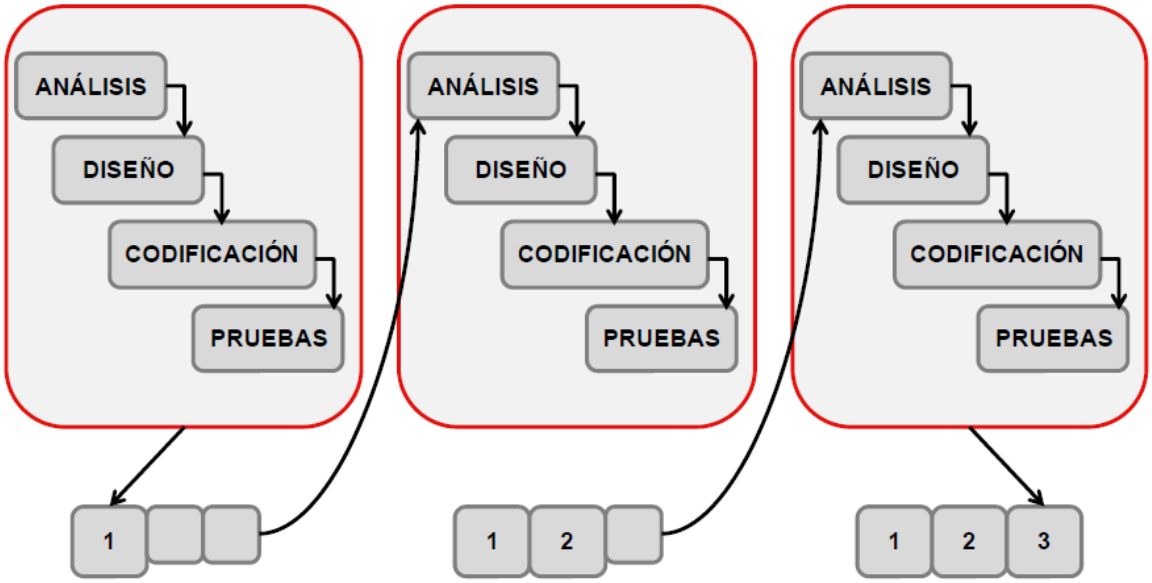

Figura 1.1 Modelo de ciclo de vida incremental [1]

Algunas de las ventajas que se obtienen al utilizar este modelo de desarrollo son:

- Generación de un software operativo de forma rápida y en etapas tempranas del ciclo de vida del software.
- Es un modelo más flexible, por lo que se reduce el coste en el cambio de alcance y requisitos.
- Es más fácil probar y depurar en una iteración más pequeña.

- Es más fácil gestionar riesgos.
- Cada iteración es un hito gestionado fácilmente.

Por otra parte las desventajas que otorga este modelo son las siguientes:

- Cada fase de una iteración es rígida y no se superponen con otras.
- Pueden surgir problemas referidos a la arquitectura del sistema porque no todos los requisitos se han reunido, ya que se supone que todos ellos se han definido al inicio.

2 Marco Teórico

2.1 Visión Humana

Se denomina visión a la capacidad que tenemos los seres vivos para interpretar el entorno que nos rodea. Según la RAE se entiende por visión a toda acción y efecto de ver [2]. El sentido de la vista es catalogado como una de las principales capacidades sensoriales que presentan los seres vivos.

El ser humano es capaz de percibir las imágenes gracias a la retina, la cual se comporta como una cámara fotográfica, que recibe las impresiones luminosas y las transmite al cerebro por las vías ópticas, los encargados de realizar esta tarea son los receptores que son denominados bastones y conos, estos minúsculos receptores se encuentran agrupados en estructuras, dentro de estas estructuras existen áreas sensibles a cada uno de los colores básicos del RGB. Una vez dentro del cerebro la información es analizada por las neuronas, altamente especializadas, distribuidas en capas. La primera capa tienen la función de detectar todos los bordes de la imagen recepcionada por el ojo, siendo esta de una gran importancia debido a que con esto se pueden interpretar los cambios en las intensidades o iluminación de la imagen, por otro lado el resto de las capas detectan los contornos y demás comportamientos que se pueden encontrar en una imagen como por ejemplo, la profundidad de esta o el movimiento [3].

2.2 Visión artificial

La visión artificial o visión computacional surge en los cincuentas, con el motivo de poder enseñarle a las computadoras a realizar tareas comúnmente asociadas a la inteligencia humana, alguna de estas con la capacidad de resolver problemas, comprender lenguajes o analizar información visual [3].

La visión artificial es una disciplina que contiene todos los procesos y elementos que le proveen la capacidad de visión a una máquina, como los ojos se lo proporcionan al ser humano. Se puede decir que la comprensión de imágenes describe la deducción automática de la estructura y propiedades de un mundo tridimensional dinámico, esto a partir de una o varias imágenes bidimensionales de ese mundo. Lo que se debe deducir del mundo tridimensional es la estructura y propiedades, no solo las propiedades geométricas como son las formas, tamaño y localización, sino también sus propiedades materiales las cuales consisten en su color, iluminación, textura y composición. Si por algún motivo se modifica el mundo que se está captando en el proceso de formación de imagen, es necesario inferir la naturaleza del cambio, e incluso predecir el futuro [4].

La visión, tanto para el ser humano como para una computadora, consta de dos fases las cuales son captación e interpretación de una imagen. La primera fase, la de captación de imagen hace mucho tiempo que está resuelta. En esta fase la cámara de video se comporta

como los ojos del computador, y su retina un sensor que es sensible a la intensidad luminosa. Por lo tanto lo que queda por realizar es la segunda fase donde la visión artificial tiene que interpretar las imágenes, distinguir los objetos de la escena, extraer información de ellos y resolver aspectos más particulares según las necesidades que se deseen satisfacer.

2.2.1 Fases de un sistema de visión artificial

La visión artificial se define en cuatro fases, las cuales tienen como objetivo poder reproducir el comportamiento humano el cual consiste en capturar la luz a través de los ojos, y que esta información sea enviada al cerebro por medio de los nervios ópticos para así procesar esta información. Existen razones para creer que el primer paso de procesado por el ser humano consiste en encontrar elementos simples en los cuales descomponer la imagen. Después el cerebro interpreta la escena y por ultimo actúa en consecuencia.

Por otra parte las fases de la visión artificial consisten en lo siguiente:

- La primera fase, es solamente sensorial, la que tiene como tarea la captura o adquisición de las imágenes digitales mediante algún tipo de sensor.
- La segunda etapa tiene como función el tratamiento de las imágenes, con objetivo de facilitar las siguientes etapas. En esta fase de procesamiento previo es donde se le aplican filtros y transformaciones geométricas a las imágenes, con la finalidad de eliminar partes indeseables de esta o destacar partes interesantes de la misma.
- La siguiente fase es la de segmentación, la cual tiene como objetivo aislar los elementos que interesan de una imagen para comprenderla.
- para concluir este proceso se encuentra la etapa de reconocimiento o clasificación. En esta se distinguen los objetos segmentados, gracias al análisis de ciertas características que se deben establecer previamente para diferenciarlos.

Las etapas anteriormente nombradas no siempre se desarrollan de una manera secuencial, sino que en ocasiones deben realimentarse hacia atrás. Así, es normal volver a la etapa de segmentación si falla la etapa de reconocimiento, o a la de pre proceso, o incluso a la de captura, cuando falla alguna de las siguientes, esto se puede apreciar en la Figura 2.1 [5].

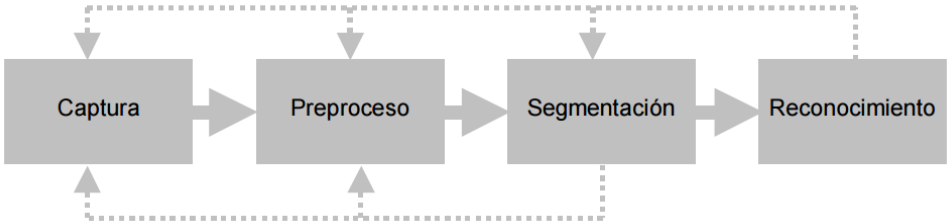

Figura 2.1 Diagrama de bloques de las etapas típicas en un sistema de visión artificial [5].

2.2.2 Componentes de un sistema de visión artificial

Para poder realizar un sistema de visión artificial (SVA), se necesita como mínimo contar con los siguientes elementos de hardware:

- Sensor óptico: Este sensor puede ser una cámara color o monocromo que produce una imagen del entorno cada 1/30 segundos. También se podría usar como sensor óptico una cámara scanner que produzca una línea cada instante. En este caso el movimiento del objeto debe ser por la línea del objeto para así poder generar una imagen bidimensional. La naturaleza del sensor y la imagen que se produce depende del tipo de aplicación a realizar.
- Tarjeta de adquisición de imagen: Esta permite digitalizar la señal de video captada por el sensor óptico.
- Computador: Es el encargado de almacenar en su memoria la imagen digitalizada, para posteriormente poder procesarla y manipularla por el programa.
- Monitor de vídeo: Permite la visualización tanto de las imágenes o escenas captadas y de los resultados del procesamiento de estas.

En la Figura 2.2 se puede observar, la iluminación, la cámara y el proceso de digitalización que permite por un lado la visualización de la imagen digital y por otro lado el proceso de extracción de parámetros y su posterior proceso de análisis y control que actúa sobre el entorno del objeto [4].

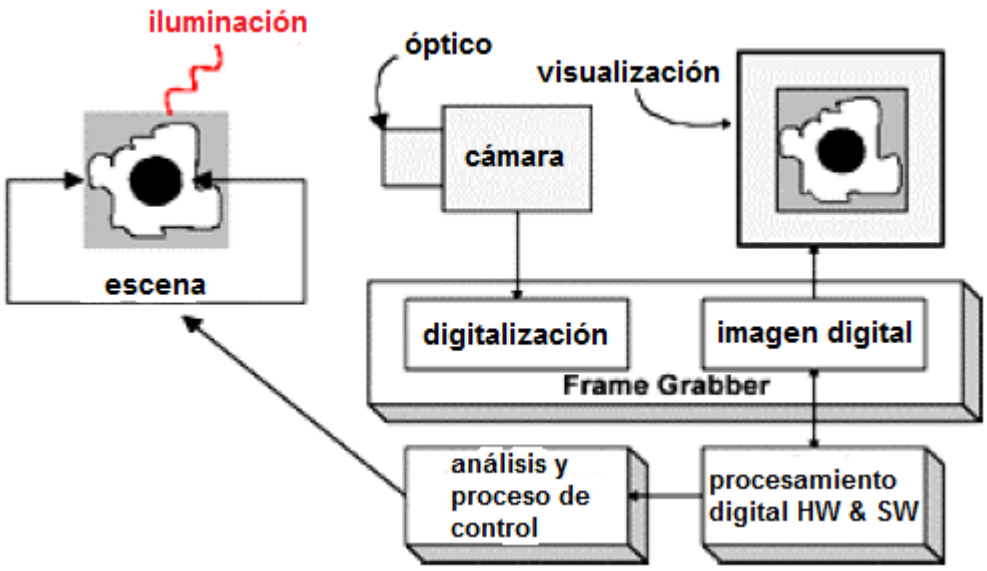

Figura 2.2 Diagrama de bloques de un sistema SVA [4].

2.3 Digitalización de Imagen

Una vez captada la imagen por la cámara se pasa a la digitalización de la imagen la cual consta de dos procesos los cuales son el muestreo y la cuantización.

2.3.1 Muestreo

El muestreo consiste en convertir una imagen continua (I_C), en una matriz discreta (I_D) de N×M píxeles. El número de muestra por unidad de espacio sobre el objeto original lleva al concepto de resolución espacial de la imagen. Esta se define como la distancia, que tienen dos pixeles adyacentes en la imagen. Sin embargo la unidad más común que se ocupa, para la medida de la resolución espacial es la de píxeles por pulgada (DPI) [5].

De esta forma, el proceso de muestreo, para una imagen, que asocia a cada punto un valor real, cambia una imagen del formato:

$$I_C (x, y) \in \Re \text{ en donde } x, y \in \Re$$

Al formato:

$$I_D (x, y) \in \Re \text{ en donde } x, y \in N \text{ y } 0 \leq x \leq N-1 \text{ , } 0 \leq y \leq M-1$$

Que se puede representar matricialmente de la siguiente manera:

$$I_D(x,y) = \begin{pmatrix} I_D(0,0) & I_D(0,1) & \dots & I_D(0,M-1) \\ I_D(1,0) & I_D(1,1) & \dots & I_D(1,M-1) \\ \dots & & & \dots \\ I_D(N-1,0) & I_D(N-1,1) & \dots & I_D(N-1,M-1) \end{pmatrix}$$

2.3.2 Cuantización

Una vez terminado el muestreó se procede a realizar la operación de cuantización, el cual consiste en la discretización de los posibles valores de cada píxel. Los niveles de cuantización frecuentemente son potencias de 2 para así facilitarle el trabajo de almacenar las imágenes al computador, debido a que estos ocupan byte como unidad mínima de memoria directamente direccionable. Por lo tanto, se suelen utilizar 2, ,4 16 o 256 niveles posibles. De esta forma, I_D que pertenece a $\Re$ se convierte en IDC (discreta cuantizada) que pertenece a N. El número de niveles posibles define la resolución radiométrica.

$$I_{DC}(x, y) \in N$$

Donde,

$$x, y \in N \text{ y } 0 \leq x \leq N\text{-}1 \text{ , } 0 \leq y \leq M\text{-}1$$

$$0 \leq I_{DC}(x, y) \leq 2^q\text{-}1$$

Si la imagen sólo tiene información sobre el brillo se dice que es una imagen en niveles de gris y se suelen utilizar hasta 256 niveles para la representación de los tonos que se encuentran desde el negro (0) hasta el blanco (255). Si la imagen solamente permite dos niveles de cuantización (normalmente banco y negro) se refiere a una imagen binaria. Para el caso del color suelen usarse 256 niveles para poder representar cada uno de los colores primarios (RGB). Con esto se obtienen 16 millones de colores aproximadamente (256x256x256) y se denomina imágenes en color real. Para los casos que se necesitan una mayor resolución radiométrica se utilizan 4096 niveles por banda de color, esto se puede apreciar en la Figura 2.3 [5].

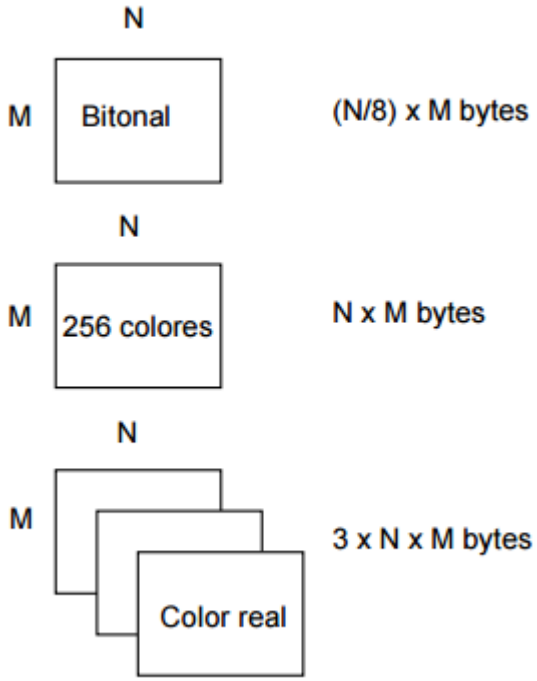

Figura 2.3 Diferentes tipos de imágenes digitales y su tamaño en bytes [5].

2.4 Procesamiento de Imagen

2.4.1 Aumento y reducción de contrastes

Antes de poder abarcar este tema se debe tener presente que el histograma de la imagen es un diagrama de barra, donde cada una de estas tiene una altura proporcional a la cantidad de píxeles que posee un nivel de cuantización. En el grafico del histograma el eje horizontal corresponde a los diferentes niveles de cuantización de valores que pueden tomar los píxeles de una imagen, mientras que el eje de las verticales corresponde al número de píxeles disponibles para cada cuantización. Para evitar la dependencia entre el número de pixeles y el tamaño del histograma, se normalizan los ejes entre 0 y 1. Debido a esto es que no salen las unidades en los ejes, en la Figura 2.4 se pueden ver dos ejemplos de histograma.

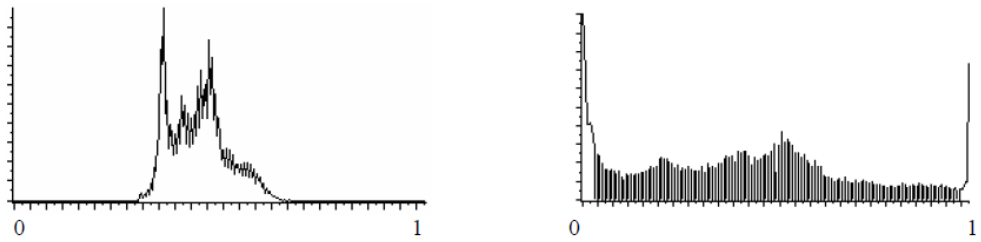

Figura 2.4 Histograma con poco contraste (izquierda), imagen saturara (derecha) [5].

Al aplicarle funciones de transferencia del histograma se pueden generar aumentos o reducción de contraste a una imagen. Estas funciones corresponden a aplicaciones, pues para cada punto del dominio solo tiene un valor imagen. Estas aplicaciones están acotadas entre 0 y 1 para ambos ejes, donde el eje horizontal corresponde a la entrada I_E del filtro y el eje vertical a la salida I_S del filtro.

La figura 2.5 muestra algunos ejemplos de funciones de transferencia: función de transferencia lineal (a) en esta no se produce ninguna modificación sobre el histograma, esto es debido a que los niveles de intensidad de la entrada y la de salida coinciden. Otro ejemplo es la función de transferencia cuadrado (b) la cual produce un oscurecimiento general de la imagen, gráficamente se puede apreciar que el rango entre 0 y 0,5 en el eje x corresponde al rango más oscuro (0 a 0,25 en el eje y). Por último la función cuadrada (c) la cual genera un aclarado general de la imagen.

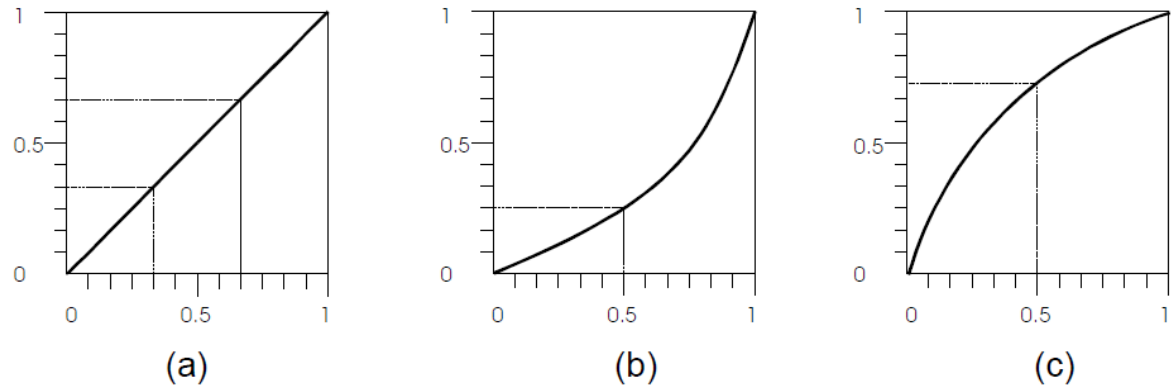

Figura 2.5 De izquierda a derecha la función lineal, cuadrada y raíz cuadrada [5].

Una definición de contraste según la RAE es: *"inexistencia o escasez de tonos intermedios, de tal manera que resaltan mucho lo claro y lo oscuro"*. Pero en este contexto se refiere a alto contraste en una imagen digital en niveles de gris si se encuentran masas separadas sobre el histograma. Un buen indicador de contraste de una imagen es el cálculo de la desviación típica de su histograma.

Para obtener una función de transferencia que consiga un efecto visual de aumentó de contraste, es decir que aclare los niveles claros y oscurezca los niveles oscuros, es necesarios componer una función de transferencia del histograma que se comporte como función cuadrada hasta el valor 0,5 y en adelante se comporte como una función raíz (a). Para generar una disminución de contraste se debería hacer lo contrario, es decir hasta el valor 0,5 se debe comportar como la función raíz y en adelante como función cuadrada (b).

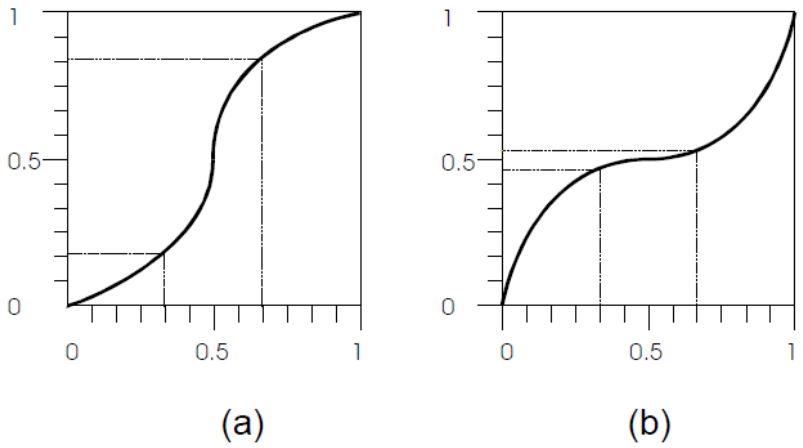

Figura 2.6 Funciones de transferencia para aumento y disminución de contraste [5].

La mayor parte de las veces, si la pendiente de la función de transferencias inferior a la unidad genera un efecto de reducción de contraste. Esto se debe a que concentra los valores de las intensidades de un rango $\mathfrak{R}$ en uno más pequeño $\mathfrak{R}$'. Mientras que el efecto de aumento de contraste se produce cuando la pendiente de la función supera a la unidad, por razones inversas a las del anterior caso.

En la Figura 2.7 se pueden apreciar los resultados cuando se aplican las funciones de transferencia anteriormente nombradas. Al aplicar el filtro de aumento de contraste se aprecia en el histograma se ha saturado los tonos claros y oscuros y que se ha reducido la densidad en la parte central de este. Lo que indica que hay muchos más píxeles asociados a los tonos blancos y negros que al resto de los tonos. Por otro lado al aplicar el filtro de reducción de contrastes se produce una imagen en la cual se puede observar una reducción del rango dinámico del histograma. Esto produce que la imagen contenga menos información que otra en que tenga un rango dinámico más amplio, ya que se homogenizó zonas de la imagen que antes eran diferentes [5].

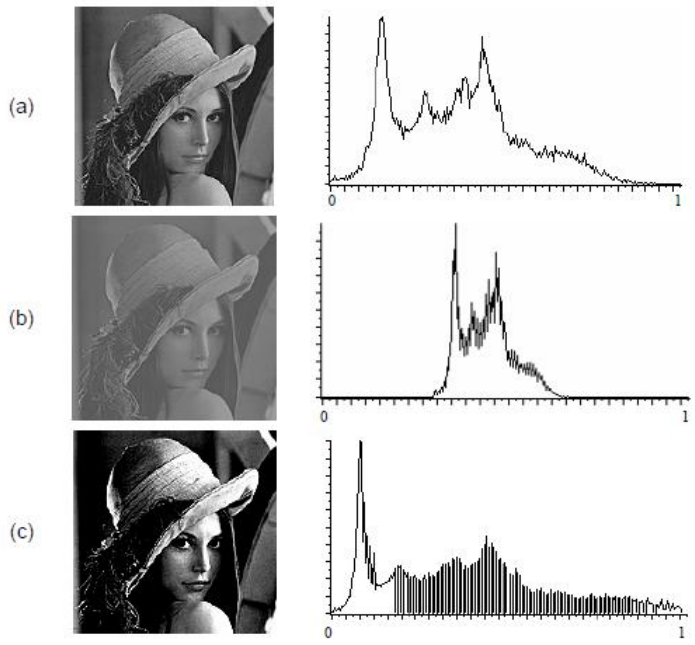

Figura 2.7 Transformaciones del histograma: (a) imagen original con su correspondiente histograma; (b) resultado de una operación de disminución de contraste; (c) aumento de contraste [5].

2.4.2 Filtrado espacial

Los filtros espaciales son filtros que se realizan directamente sobre la imagen, con el fin de resaltar o atenuar detalles espaciales con el propicio de mejorar la interpretación visual y facilitar un procesamiento posterior. El procedimiento de esto consiste en recorrer toda la imagen, píxel a píxel, realizando alguna operación aritmética sobre estos, para que así su valor sea modificado de acuerdo con los valores de los píxeles que lo rodean, con esto se trata de transformar los niveles de gris originales de tal forma que se parezcan o diferencien más de los píxeles vecinos, al conjunto de estos vecinos se les denomina ventana, los cuales tienen una dimensión impar como 3X3, 5X5, etc. El filtro más usado de este tipo es el filtro espacial de convolución [4, 5, 6].

Se denomina convolución a una función, que de forma lineal y continua transforma las señales de entradas en señales de salida. Esto lo hace sumando una función f consigo misma repetidamente veces en todo el dominio de otra función h, la cual utiliza en cada suma como valor de escala el valor de h en ese punto de su dominio. Su fórmula matemática es la siguiente [5, 7]:

$$f(x) * h(x) = \int_{-\infty}^{\infty} f(x)h(u-x)dx$$

La operación de convolución tienen la propiedad de conmutatividad, pero en la teoría de filtros las funciones corresponden a la señal del entrada y la otra al filtro que se le va a aplicar a la imagen, a esta última se le denomina función impulsional.

En el caso bidimensional discreto, la operación de convolución se define mediante la siguiente ecuación:

$$I'(x,y) = I(x,y) * h(x,y) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} I(i,j) \cdot h(x-i, y-j)$$
$$\forall x, y = 0,1,..., N-1$$

Donde I es una matriz de tamaño N y h es una función periódica de periodo N. En este caso de una imagen de tamaño NxN, la transformación inversa (h) del filtro tiene como resultado una matriz también de tamaño NxN, la cual se conoce como función de filtrado espacial o matriz o mascara de convolución.

En principio la carga computacional de esta operación es elevada si N es grande. Sin embargo es posible que la función de transferencia h tenga un número de elementos muy inferior a NxN al estar muchos de sus valores a cero. Utilizando una función de convolución de tamaño menor al de la imagen obtenemos la siguiente expresión:

$$I'(x,y) = \frac{1}{D} \sum_{i=0}^{n} \sum_{j=0}^{n} I(i,j) \cdot h(x-i, y-j) \qquad \forall x, y = 0,1,..., N-1$$

Donde D es un factor de escala que se conoce como factor de división [5].

2.4.3 Filtro de suavizado

Este tipo de filtrado se ocupa generalmente para reducir los efectos negativos que se presentan en un imagen al momento de ser capturada por el computador. Este consiste en sustituir los píxeles de una imagen por la media aritmética de los puntos que tiene alrededor un pixel, incluido el mismo. Para esto se toma una ventana de tamaño NxN, donde el centro de esta matriz es el punto a sustituir, se calcula la media de la suma de todos los valores de los píxeles de la ventana y se sustituye el pixel en la nueva imagen por el valor obtenido. La precisión de este filtro aumenta si el tamaño de la ventana es mayor.

Los resultados que entrega este filtro corresponden a una difuminación de la imagen produciendo un suavizado de los gradientes en toda la imagen. Este tipo de filtro es equivalente a un filtro promediador de área [4].

Figura 2.8 Aplicación de un filtrado espacial de suavizado [5].

2.4.4 Filtros de obtención de contornos

Para poder determinar los cambios en una dirección determinada sobre una imagen, se utiliza el cálculo de la derivada direccional. Estos cambios suelen pertenecer a los contornos de los objetos en la imagen que se está procesando.

Partiendo de que el operador gradiente se define como:

$$\nabla(I(x,y)) = \frac{\partial I}{\partial x}\vec{u}_x + \frac{\partial I}{\partial y}\vec{u}_y$$

Se definen los filtrados de convolución G_x, y G_y como:

$$G_x = \frac{\partial I}{\partial x} = I(x,y) * h_1(x,y)$$

$$G_y = \frac{\partial I}{\partial y} = I(x,y) * h_2(x,y)$$

$$h_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix} \qquad h_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{pmatrix}$$

Con la primera matriz (h1) proporciona un filtrado en el que un cambio de brillo entre dos píxeles adyacentes en horizontal produce un valor distinto de cero. Mientras que la segunda matriz (h2) proporciona la misma información pero en este caso en píxeles adyacentes verticalmente. Los cambios de claro a oscuro son representados con un valor positivo y los cambios de oscuro a claro con un valor negativo. Mientras que si los píxeles adyacentes tienen igual valor de convolución, ese punto devuelve cero.

Las siguientes matrices son conocidas como ventanas de Sobel. Mediantes ellas se calcula el gradiente en las direcciones horizontales y verticales.

$$h_1 = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \qquad h_2 = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

En la Figura 2.9 se muestra una imagen a la cual se le aplicó h1, en esta imagen de resultado se pueden apreciar los contornos horizontales de la imagen original. Ese resultado se obtiene utilizando un factor de división 4 y presentando un valor absoluto de la convolución, utilizando niveles de gris en escala desde 0 (blanco) hasta 255 (negro).

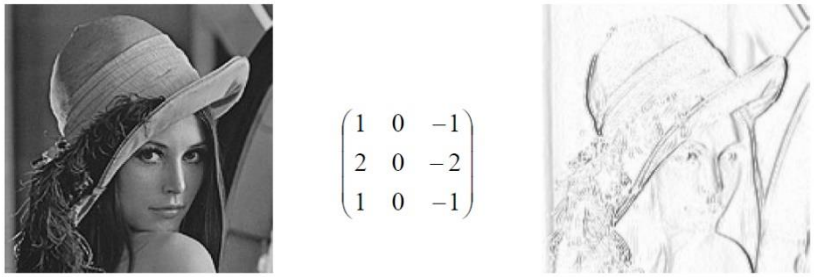

Figura 2.9 Filtrado de Sobel en la dirección x en valor absoluto [5].

Para evitar los valores fuera de rango, comúnmente se utiliza el valor absoluto. La cual consiste en el uso de un factor de suma que se le aplica luego de la convolución y la división. En un filtrado Sobel se podría utilizar un factor de división 8 y un factor de suma de 128 para evitar valores fuera de rango [5].

2.5 Open CV

OpenCV es una librería de visión artificial con licencia BSD y por lo tanto es gratis, tanto para el uso académico y comercial. Es multiplataforma, posee versiones para Linux, Mac OS X y Windows. Abarca más de 500 funciones con las que permite el proceso de visión, reconocimiento de objetos, calibración de cámara, visión estérea y visión robótica. A su vez cuenta con interfaces de C, C++, Python y java (JavaCV) [8,9].

OpenCV incluye cuatro componentes diferentes para la visión por computador en tiempo real. El primer componente es CxCore, el cual proporciona los tipos genéricos que se utilizan por las demás librerías como pueden ser árboles, listas, colas, secuencias, imágenes y las funciones que operan sobre estos. Otro componente es CvReference que incluye las funciones para el análisis y procesamiento de los datos. También esta CvAux, este contiene

componentes experimentales y obsoletos de OpenCV. Como último se tiene HighGUI que nos proporciona los elementos necesarios para poder capturar, mostrar y almacenar imágenes.

2.6 Agentes Inteligentes y Sistema Multi-agente.

2.6.1 Agente.

La definición de agentes puede ser variada, esta depende de la situación en la cual se esté definiendo. Para Russel y Norving un agente es "Toda entidad que de forma autónoma percibe su entorno (real o simulado) mediante sensores y que actué en el mismo mediante efectores" [14], mientras que Wooldridge define agente como "Un sistema informático que se sitúa en un cierto ambiente y que es capaz de acción autónoma en este ambiente para lograr sus objetivos de diseño" [15]. De estas definiciones se puede concluir, que un agente es capaz de actuar de forma autónoma dentro de un entorno real o simulado, para lograr sus objetivos.

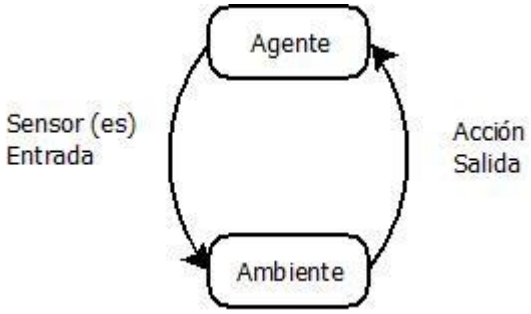

Figura 2.10 Agente y su Ambiente [elab. Propia]

2.6.2 Agente Inteligente.

Al igual que la definición de agente, al buscar una definición de Agente Inteligente se pueden encontrar diferentes descripciones de esta. Para Wooldridge un agente inteligente "es una entidad software que, basándose en su propio conocimiento, realiza un conjunto de operaciones para satisfacer las necesidades de un usuario o de otro programa, bien por iniciativa propia o porque alguno de éstos se lo requiere" [15]. Al tomar esta definición, se puede entender que todo agente inteligente es un programa, pero no todo programa que realiza una búsqueda es un agente inteligente.

Para que un agente sea considerado un agente inteligente debe poseer las características de Reactividad, Pro-actividad y Habilidad social. A continuación se definen estas tres características y algunas otras características que posee un agente inteligente.

- Autonomía: Los agentes no necesitan la intervención de un humano para operar.

- Habilidad Social: Tiene la capacidad de interactuar con otros agentes por medio de un lenguaje de comunicación de agentes.
- Racionalidad: Razonan acerca de los datos obtenidos con el fin de llegar a una solución óptima.
- Movilidad: Pueden moverse a través de una red.
- Veracidad: un agente no comunica información falsa de manera deliberada.
- Benevolencia: Los agentes se ayudan mutuamente si esto no se opone a su objetivo.
- Reactividad: Los agente perciben estímulos del entorno, los cuales los guían las acciones de su entorno.
- Pro-activo: No solo reaccionan por estímulos, también pueden actuar guiados por sus objetivos.

2.6.3 Sistema Multi-agente

Los agentes inteligentes son usados para solucionar problemas, pero cuando este es de una gran complejidad es necesario más de un agente para poder encontrarle una respuesta al problema. Cuando sucede esto el problema se divide en problemas de menor complejidad y de más fácil solución, donde cada uno de estos problemas se les asigna a diferentes agentes. A esto se le llama sistema multi-agente (SMA) [13, 15].

Los SMA son un conjunto de agentes inteligentes que resuelven problemas y trabajan juntos para encontrar respuesta a problemas que sobrepasan sus capacidades individualmente. Actualmente su área de investigación es muy activa y se está utilizando para aplicaciones del tipo comercial e industrial. Estos se centran principalmente en el comportamiento social de entidades inteligentes y del estudio de modelos de comportamiento, estrategias de cooperación y coordinación, mejorar el desempeño de tareas, entre otros [13, 15].

Un SMA se caracteriza por:

- Estar formado por un conjunto de agentes, donde cada uno de ellos mantiene sus habilidades de adquisición de datos, comunicación, planificación y actuación.
- Tener una misión común para todo el sistema, esta se puede descomponer en múltiples tareas diferentes de manera que se puedan ejecutar en paralelo. A su vez, se le debe asignar a cada agente una o varias tareas teniendo en cuenta el objetivo final.
- Los agentes del sistema tienen un conocimiento limitado, esta limitación puede ser del conocimiento del entorno, de la misión del grupo o de las intenciones de los demás agentes a la hora de realizar sus propias tareas.
- Cada agente tiene cierta especificación para realizar las tareas asignadas, en función de lo que conoce, la capacidad de procesamiento y la habilidad requerida.

Distribuir las tareas de un sistema en diferentes agentes permite:

- Mantener la autonomía de cada agente. Cada uno de los agentes decide en función de su propio entorno las tareas a llevar a cabo.
- Distribuir la información del sistema en diferentes agentes, para no tener la necesidad de contener toda la información del sistema en un solo agente.
- Descentralizar la toma de decisiones, para permitir a los agentes tomar decisiones que involucran a otros agentes. Con esto hace que el sistema sea más robusto a posibles fallos.
- Llevar a cabo acciones coordinadas, para realizar esto es necesario tener un sistema de comunicación entre los agentes.
- Organizar dinámicamente la arquitectura de agentes, para permitir la adaptación de esta a posibles cambios que se produzcan en el entorno.

2.6.4 Infraestructura para agentes

Para poder realizar lo ya nombrado con anterioridad, respecto a la comunicación entre agentes para poder conseguir el objetivo deseado del sistema, es necesario contar con la infraestructura de agentes. Esta proporciona las regulaciones que los agentes siguen para comunicarse y para entenderse [17].

Esta infraestructura se compone de los siguientes elementos:

- **Ontologías:** Es un significado común para un conjunto de símbolos o términos para los agentes, para que se puedan entender. Cuando dos agentes utilizan la misma ontología se aseguran de que sus interpretaciones de lo que se están comunicando coincidan.
- **Lenguajes de Comunicación (ACLs):** Define la estructura que debe tener los mensajes que son enviados por los agentes a los demás agentes, permitiendo así el intercambio y entendimiento de mensajes entre estos. Dentro del ACL nos encontramos tres aspectos, los cuales son la sintaxis, la cual estructura la simbología de la comunicación, la semántica, encargada de la denotación de los símbolos y pragmática, la que interpreta los símbolos.
- **Protocolos de Interacción:** Es el encargado de describir el patrón de comunicación, para permitir la interacción entre agente, permitiendo la conversación entre agentes, la que consiste en intercambios estructurados de mensajes [17].

2.6.5 Comunicación entre agentes

Una de las definiciones de la RAE respecto a comunicación es “Transmisión de señales mediante un código común al emisor y al receptor”. Para realizar esto se tiene que tener en cuenta que el emisor crea un mensaje, que contiene lo que se desea transmitir, además a este se le debe dar un sentido y finalmente, se encuentra la acción que realiza el receptor una vez que recibe el mensaje. En los sistemas multi-agente también se tiene que tener en cuenta estos aspectos para permitir la comunicación entre dos o más agentes.

Lo ya mencionado en SMA se conoce como *Speech Act Theory*, con esto se consigue dar sentido a los mensajes mediante el concepto de performativa, permitiendo registrar la semántica del acto comunicativo. Algunos tipos de performativa para dar sentido a los mensajes de un agente pueden ser un requerimiento, una declaración, una pregunta, entre otras.

Los lenguajes de alto nivel de comunicación entre agentes, incorporan esta teoría. Estos están fundamentalmente constituidos por una serie de performativas, los cuales expresan los diferentes actos comunicativos que pueden realizar un agente. Junto con esto surgen la necesidad de estandarizar la sintaxis y la semántica del lenguaje que utilice el agente para comunicarse con otro. Las propuestas más importantes en esto son KQML (Knowledge Query and Manipulation) y FIPA-ACL (Foundation for Intelligent Physical Agents - Agent Communication Language) [18].

Con los lenguajes mencionados, permite a los agentes construir un mensaje y aportar sentido al acto comunicativo entre un agente emisor y receptor. Pero también se le debe expresar contenido a este mensaje de tal manera que pueda ser entendido por los agentes involucrados en la transmisión de mensajes. Por eso es necesario un lenguaje el que permita expresar descripciones del dominio de forma entendible por los agentes involucrados. Para esto existen lenguajes estandarizados de contenido de tipo lógico como KIF (Knowledge Interchange Format) y FIPA-SL (FIPA Semantic Language) [18].

Por otra parte los agentes involucrados en la comunicación deben emplear un vocabulario común y que el contenido del mensaje para ambos actores signifique lo mismo. Este vocabulario común viene a expresar la necesidad que los agentes posean la misma ontología [18].

2.6.6 Metodología PASSI

PASSI (Process for Agent Societies Specification and Implementation) es una metodología para poder desarrollar proyectos orientados a agentes. El objetivo de éste es poder desarrollar y diseñar una sociedad multi-agente por medio de modelos de diseño y conceptos de ingeniería de software orientada a objeto y sistemas multi-agente usando notación UML. Esta metodología está compuesta por cinco niveles de diseño y doce pasos en el procedimiento de construcción de un sistema multi-agente [19].

Modelo de Requerimientos del Sistema: Modelo antropomórfico de los requerimientos del sistema en término de agentes y objetivos. Se compone de:

- Descripción de Dominio: Descripción funcional del sistema utilizando casos de uso convencionales.
- Identificación de Agentes: Separación de responsabilidades entre agentes, representados por paquetes UML.

- Identificación de Roles: Uso de diagramas de secuencia para explorar las responsabilidades de cada agente a través de escenarios con roles específicos.
- Especificación de Tareas: Especificación por medio de diagramas de actividad de las capacidades de cada agente.

Modelo de Sociedad de Agentes: Modelo por el cual se pueden expresar las interacciones sociales y dependencias entre los agentes involucrados en la solución. Este modelo se compone de:

- Identificación de Roles: Se realiza igual que en el Modelo de Requerimientos del Sistema.
- Descripción de Ontología: Uso de diagrama de clases y restricciones de objetos (OCL) para describir el conocimiento utilizado por cada agente y la pragmática de sus interacciones.
- Descripción de Roles: Uso de diagramas de clase para mostrar los distintos roles que cumplen los agentes, las tareas involucradas dependiendo de sus roles, capacidades de comunicación y dependencias entre agente.
- Descripción de Protocolos: Uso de diagramas de secuencia para especificar la gramática de cada protocolo de comunicación en términos de performativas.

Modelo de Implementación de Agentes: Modelo de la arquitectura de la solución en términos de clases y métodos. Cual se construye por:

- Definición de Estructura de Agentes: Uso de diagramas de clase convencionales para describir la estructura de las clases agente utilizadas en la solución.
- Descripción del Comportamiento de Agentes: Uso de diagramas de actividad o diagrama de estado para describir el comportamiento individual de los agentes.

Modelo del Código: Modelo de la solución a nivel del código, compuesto por:

- Reutilización de Código: Librería de diagramas de clase y de actividad con el código reutilizable asociado.
- Código Completo: Código fuente del sistema objetivo.

Modelo de Despliegue: Modelo de la distribución de las partes del sistema a través de unidades de procesamiento de Hw y su migración entre unidades de proceso. Involucra un paso:

- Configuración de Despliegue: Uso de diagramas de despliegue para describir la colocación de agentes en las unidades de procesamiento disponibles y cualquier restricción a la migración o movilidad [19].

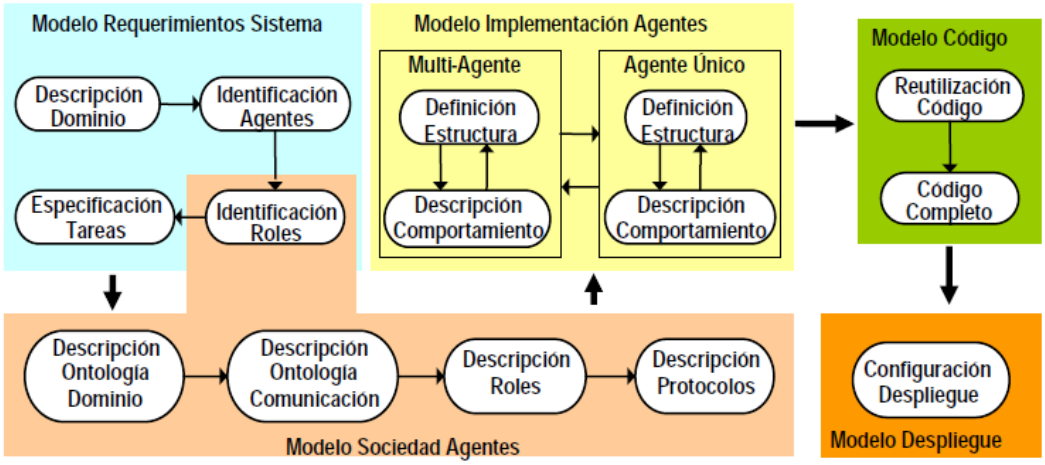

Figura 2.11 Esquema de modelos de fases PASSI [19]

2.6.7 Plataforma JADE

Java Agent Development Framework (JADE), es un middleware para el desarrollo de agentes y desarrollo de sistemas multi-algentes bajo el estándar FIPA, implementado en Java. Además proporciona una implementación estándar del lenguaje de comunicación FIPA-ACL, que facilita la comunicación entre agentes y permite la detección de servicios que se proporcionan en el sistema. Este fue desarrollado originalmente por Telecom Italia y se distribuye como software libre [16].

Adicionalmente JADE proporciona:

- Un entorno de ejecución en donde los agentes de JADE se ejecutan.
- Bibliotecas de clases para la creación de agentes mediante la herencia y la redefinición de comportamientos.
- Un conjunto de herramientas gráficas para la monitorización y administración de la plataforma de agentes.

2.7 Robótica

Según la RAE, la palabra robótica es “una técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones industriales”. Actualmente la mayor parte de los robots que se construyen, se encuentran en fábricas, donde se utilizan para realizar las tareas más pesadas o repetitivas para los seres humanos, como lo son soldar, ensamblar, transportar objetos, entre otras.

Las personas ven los robots como un medio para poder facilitar algunas de las tareas pesadas o tediosas del día a día. Es por esta razón que se ha demostrado un gran interés en esta

área, esto se puede ver demostrado por los avances que se han conseguido, a través del trabajo de androides. Estos consisten en ser un robot con apariencia humana, intentando simular lo más posible a una persona.

Un ejemplo de androide, es el robot experimental Wabot-1 (figura 2.12) creado en Japón, este fue usado por más de una década en la experimentación de movilidad y reacción ante objetos que lo rodean. Wabot puede caminar, bajar y subir escaleras y mover objetos, esto por medio de sensores, cámaras, micrófonos y altoparlantes que posee.

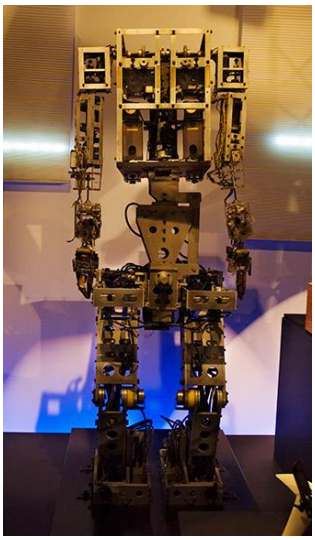

Figura 2.12 Wabot-1[20]

La robótica también es utilizada en diferentes áreas de estudio, una de estas áreas es la medicina donde existe Sim One. Este es un modelo de ser humano que es controlado por computador. El fin de este androide es poder ayudar a los estudiantes de medicina, debido a que posee un corazón que late, respira y sus ojos parpadean al igual que una persona. A demás puede simular diferentes enfermedades y responder a diferentes medicinas y tratamientos para un mejor estudio de los estudiantes.

También existe otro tipo de robot, los cuales son usados a control remoto para poder llegar a lugares y situaciones peligrosas para los seres humanos, como lo son exploraciones submarinas, inspección de construcciones subacuáticas e instalaciones petroleras, exploración de otros planetas, entre otros. Algunos ejemplos de esto son Consub que permite descender a grandes profundidades en el mar y Sojourner que es el robot de exploración utilizado en la misión espacial PathFinder, el cual fue enviado al planeta Marte para recoger información de la superficie de este [20].

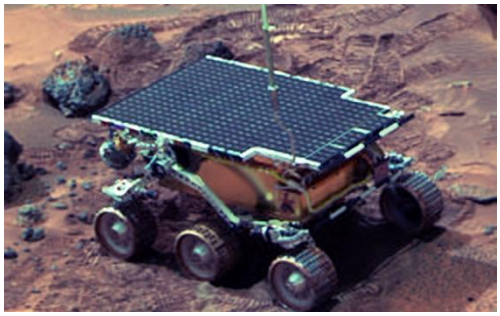

Figura 2.13 Sojourner[20]

2.8 Lego Mindstorms

Lego Mindstorms es un kit de robótica fabricado por Lego, este posee elementos básicos de la teoría robótica, como lo son la unión de piezas y programación de acciones, de forma interactiva. Su primera comercialización fue bajo el nombre de Robotic Invention System, en septiembre de 1998. Actualmente existen dos versiones de Mindstorms una comercial y otra educacional. Esta última versión se vende como herramienta educacional, la cual viene con un software programable basado en la GUI de Robolab. Con Mindstorms prácticamente todo puede ser representado con las piezas tal como en la vida real. De ahora en adelante se nombrada como modelo NXT, el cual será usado en el desarrollo del proyecto [22].

2.8.1 Ladrillo NXT

El ladrillo NXT es una versión sucesora de la RCX, que se considera la predecesora y precursora de los bloques programables de Lego. La diferencia en la versión educacional con la comerciable de esta edición, es que incluye baterías recargables y el cargador, pero se debe comprar el software según el tipo de licencia que se desea ocupar Personal, Sala de clases, Sitio [21].

El bloque NXT es el cerebro de los proyectos que se realizan con Mindstorms. En su interior tiene un microcontrolador programable, dentro del cual se encuentra el programa necesario para que los motores funcionen de acuerdo a lo ordenado [22].

Figura 2.14 Ladrillo Lego Mindstorms NXT [22]

2.8.2 Microcontrolador

El microcontrolador que contiene el ladrillo NXT es un ARM7 de 32 bits, que incluye 256 Kb de memoria Flash y 64 Kb de RAM externa, la cual a diferencia del bloque RCX, posee mayores capacidades de ejecución de programas, evitando que los procesos inherentes de varios paquetes de datos colisionen y produzcan errores y un posible error en la ejecución del software [22].

2.8.3 Servomotores

Estos permiten la detección de giros de la rueda, indicando los giros completos o medios giros, que es controlado por el software. Estos poseen dos funciones la primera consiste en determinar la velocidad o ángulo de movimiento del robot y también actúan como sensores de rotación [21, 22].

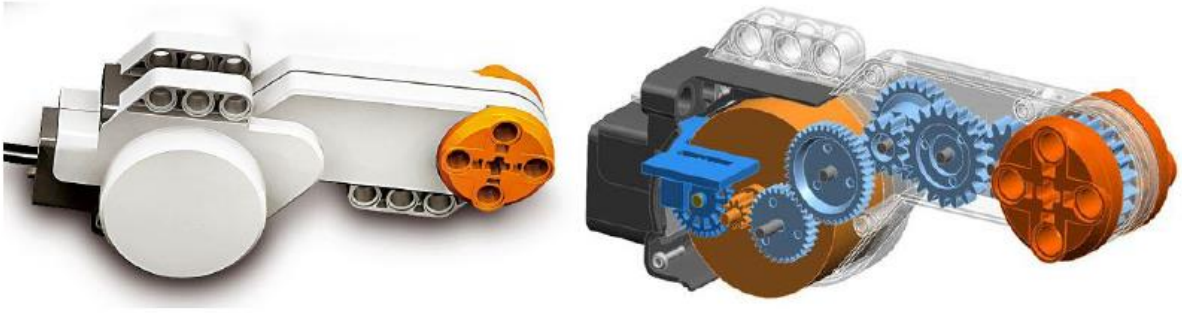

Figura 2.15 Servomotores [22]

2.8.4 Sensores

Los robots NXT constan de diferentes sensores para poder interactuar con el ambiente que les rodea; algunos de estos son:

- **Sensor de Sonido:** Incorpora un pequeño micrófono, permitiendo así que se pueda programar un robot que al escuchar un ruido realice alguna acción.
- **Sensor de Contacto:** Permite detectar si el bloque que lo posee este sensor ha colisionado o no con algún objeto que se encuentre en su trayectoria inmediata.
- **Sensor de Distancia:** Permite medir distancias de 7 a 1,80 metros, aportándole al robot una capacidad de visión parecida a la de los murciélagos, a través de ultrasonidos.
- **Sensor de Luz:** Le permite al robot distinguir entre luz y oscuridad. Con esto se le da un sentido de visión a este.
- **Sensor Brújula:** Este sensor permite medir el campo magnético de la tierra y calcula una dirección magnética con la que guiar el sistema de dirección de tu robot.
- **Sensor IRSeeker V2:** Usado para detectar la bola (IRBall HiTechnic) emisora de Infrarrojos, posee cinco detectores internos, con una visión de 240.

Figura 2.16 Sensores [elab. Propia]

2.9 LeJOS NXJ

LeJOS es un reemplazo para el firmware original del NXT, actualmente se puede instalar en los ladrillos de RCX y de NXT. Con este se puede programar los mindstorms en lenguaje Java. Incluye una máquina virtual de Java, la cual permite ejecutar las aplicaciones que se han desarrollado bajo este lenguaje [21].

LeJOS además de ser un firmware, ofrece un conjunto de utilidades algunas como lo son:

a) Una librería de clases Java que implementan la API de este.
b) Un enlazador el cual permite crear archivos binarios para poder ser cargados en el ladrillo.

c) Una herramienta para el PC para flashear el firmware en el NXT, programa para cargar aplicaciones, debugger, entre otras funciones.
d) Una API para poder desarrollar programas para el computador y que estos se puedan comunicar con el NXT por medio de conexión Bluetooth, USB o usando LEGO Communications Protocol.

3 Desarrollo del proyecto

3.1 Finalidad del proyecto

La finalidad de este proyecto, es poder desarrollar un sistema multi-agente, con el cual se pueda detectar la posición y dirección en la cual se encuentra una Lego NXT dentro de un escenario, como una especie de GPS, para así poder complementarlo con otros proyectos, los cuales ocupen este tipo de robot. Debido que al tener las posiciones en las cuales se encuentra se puede utilizar de forma aleatoria la posición de inicio, también se pueden prevenir que estos colisionen entre sí, entre otras ventajas que se pueden tener al conocer su posición.

La posición y dirección se desea obtener por medio de una cámara, la cual estará a una altura determinada por el tamaño del espacio en cuales se podrán mover los NXT. La cámara estada conectada al computador, donde se captara una figura geométrica que se encontrara sobre el NXT, para su procesamiento y así poder enviar la posición en cual se encuentran estos y evitar los problemas ya antes nombrado. La figura más óptima para este procedimiento es el triángulo isósceles, debido a que la detección de su base es simple de detectar.

Algunos de los problemas que se pueden identificar para el desarrollo de este proyecto son, la comunicación entre los agentes, debido a que en el transcurso de enviar y recibir mensajes entre un agente y otro existe una diferencia de tiempo entre él envió y la recepción. Esto provoca que si el robot se encuentra en movimiento la posición en la cual se encuentra al momento que se recibe el mensaje ya no corresponda a la del momento en que se envió el mensaje. Lo ya mencionado a su vez provoca problemas en obtener la localización y orientación del robot.

Otro problema que se puede observar es la coordinación entre los NXT, debido a que es una de las partes fundamentales del proyecto, al ocurrir una descoordinación puede causar que el sistema completo falle. También la interacción con el ambiente es uno de los problemas que se pueden observar, puesto que un simple cambio en este puede generar un escenario adverso. Con esto se pueden producir problemas en la realización de las tareas que debe realizar el Mindstorms o en la localización de este.

Una vez encontrado estos problemas, se analizarán y se les buscará la solución óptima posible. Para esto el proyecto se dividirá en sub proyectos para encontrar la solución. Primero se estudiada la detección de la figura geométrica por medio de OpenCV, debido a que es la parte fundamental del proyecto, por que con esta se obtendrá la posición en cual se encuentra cada uno de los Lego. Una vez solucionado esto se procederá a realizar un sistema multi-agente para permitir la detección de un NXT por medio del sistema desarrollado en OpenCV. En este mini sistema se probara que el robot no se salga del campo que se le asigno para moverse. Una vez finalizado esto se iniciara con el desarrollo de un sistema multi-agente que permita solucionar el objetivo de este proyecto, escalando lo ya realizado para que se pueda trabajar con más de un robot a la vez.

El escenario final del proyecto consistirá en una cámara la cual estará a una altura en la cual pueda abordar en su totalidad el terreno en cual se moverán los NXT. La cámara estará

conectara a un computador para recibir las señales de la cámara y tener funcionando el sistema multi-agente. Este sistema le mandara por medio de Bluetooth al robot todas las indicaciones que debe realizar para cumplir el objetivo del sistema. A continuación se muestra un esquema de cómo será el escenario final.

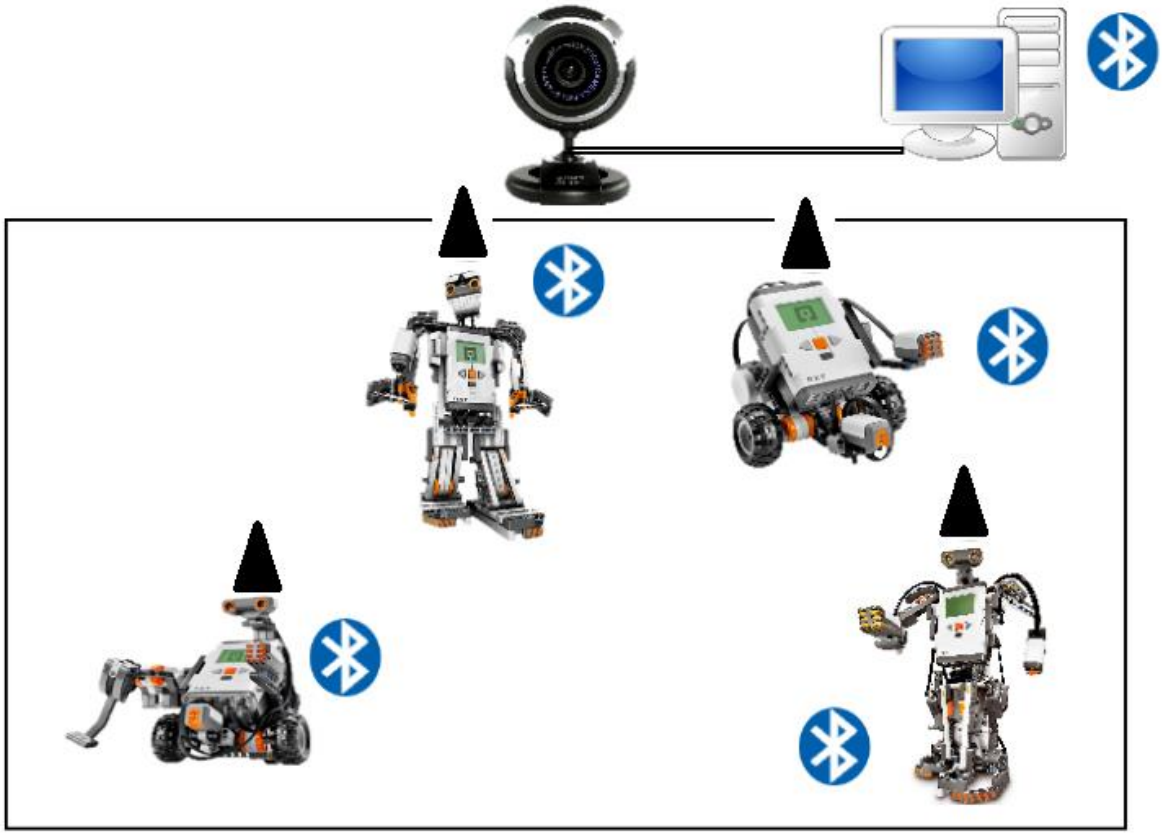

Figura 3.1 esquema escenario Final

3.2 Desarrollo del Modelado del Sistema Multi-Agente.

En esta entrega el modelado del sistema multi-agente se desarrollada bajo la metodología PASSI explicada en punto 2.6.6.

3.2.1 Modelo de Requerimientos del sistema.

Con los actuales modelos se pretende tener una visión general de lo que el sistema SMA debe realizar. Esta visión se obtiene a través los diagramas de casos de uso, diagrama de secuencia y actividad, con los que se pueden observar las funciones, roles y tareas que debe realizar el sistema.

3.2.1.1 Descripción del Dominio

Por medio de este modelo se pueden identificar todas las tareas que debe desarrollar el sistema multi-agente, para tener un funcionamiento óptimo. Aunque aún no se identifican los agentes que se utilizaran en este diagrama se pueden apreciar las tareas que tiene que llevar acabo el sistema, como lo son la gestión del suscriptor, las que se encargan de realizar todas las tareas que debe realizar el sistema al que se le quiera incorporar el sistema de posicionamiento y direccionamiento. Otra tarea es la gestión del localizador encargado de captar la posición y dirección de los NXT.

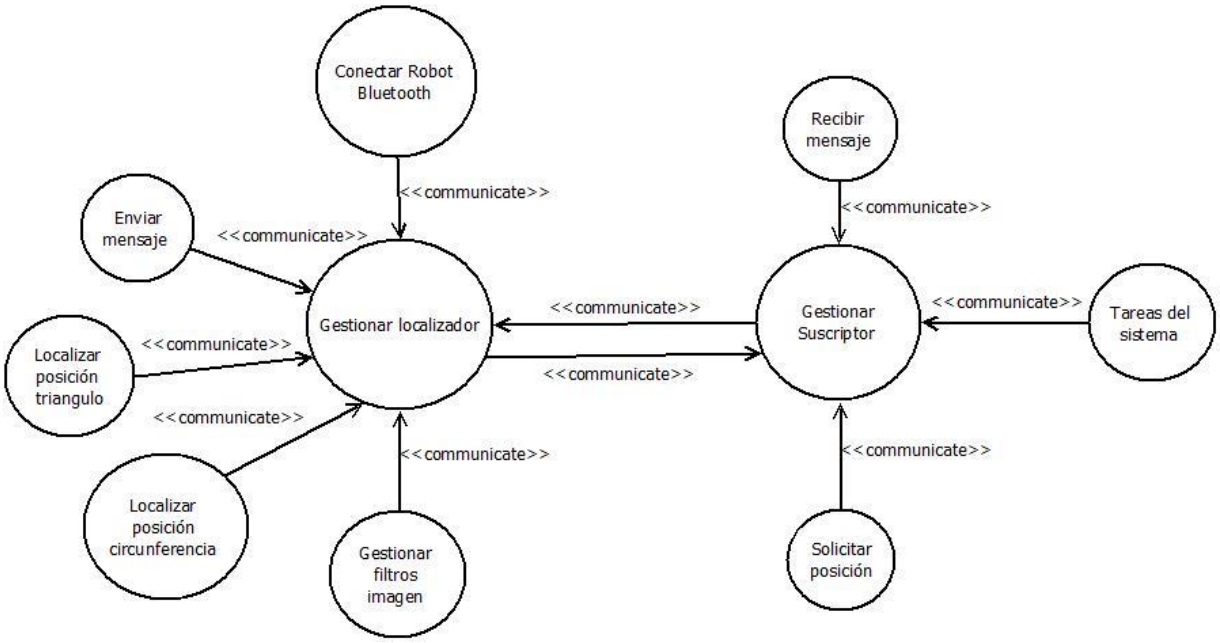

Figura 3.2 Descripción de dominio

3.2.1.2 Identificación de agentes

Por medio del Diagrama de descripción del dominio, se identifican los agentes que se utilizaran en el sistema. Para esto se realizan paquetes para agrupar las actividades de cada agente.

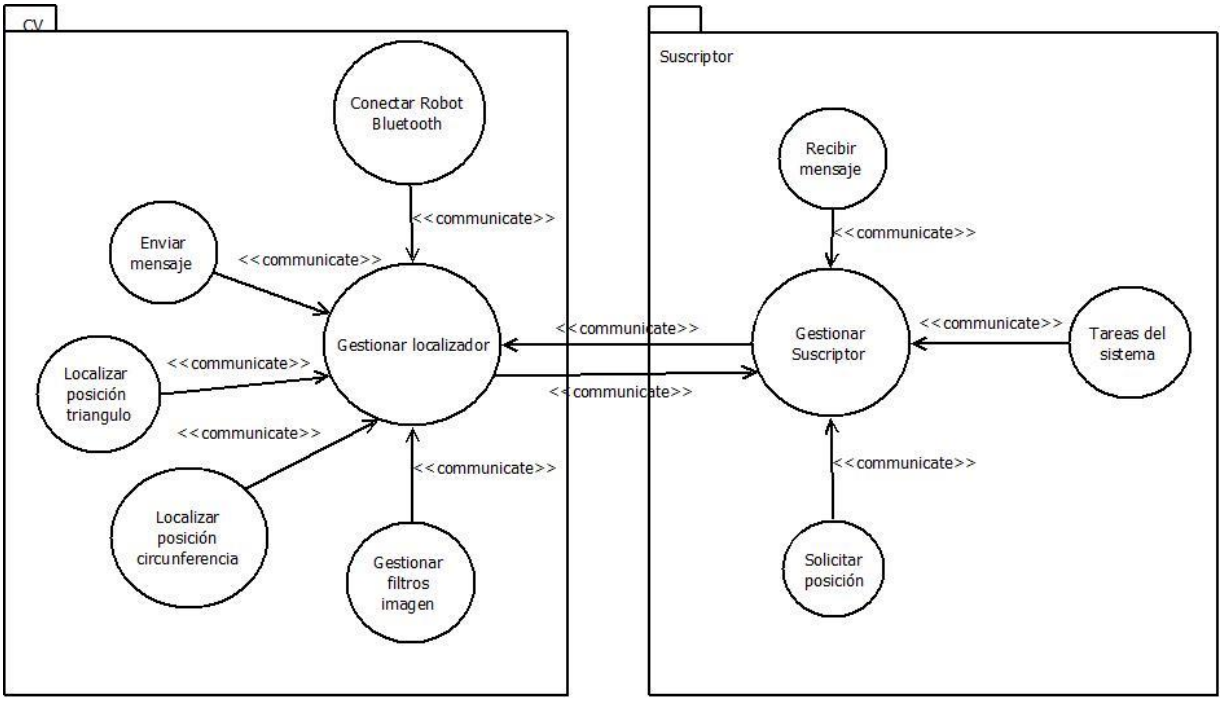

Figura 3.3 Identificación de Agentes

Los agentes identificados son:

- CV: Es el encargado de localizar al robot dentro del campo.
- Suscriptor: Es el encargado de realizar las tareas del sistema a cual se le desea implementar el sistema de posicionamiento y direccionamiento.

3.2.1.3 Identificación de roles

El diagrama muestra los roles que interactúa en el funcionamiento del sistema. Se puede observar que el AgenteCV actúa como Broker. Este se encarga de comunicar las acciones que se deben realizar y en qué momentos realizarlas, para así poder llegar al objetivo del sistema.

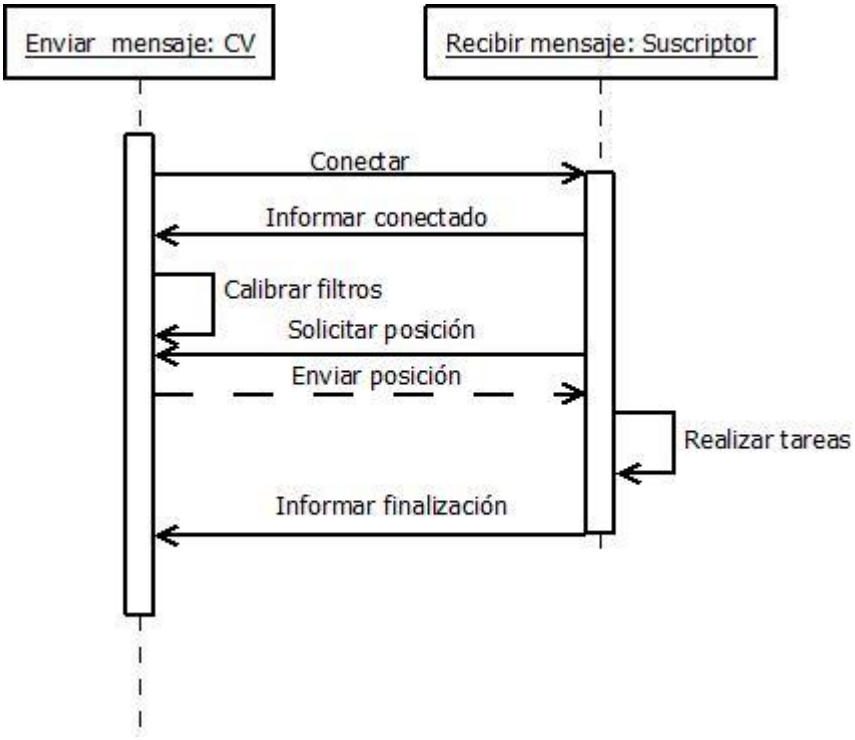

Figura 3.4 Identificación de Roles

3.2.1.4 Especificación de Tareas

El siguiente diagrama muestra la forma como se comportan los agentes para llegar a cumplir el objetivo del sistema. En este se puede observar que el AgenteCV se conecta al sistema suscriptor, una vez conectado este calibra el filtro de la imagen. Completando los pasos ya mencionados el suscriptor solicita su posición, el AgenteCV le envía si posición una vez captada, para que así poder realizar las tareas del sistema.

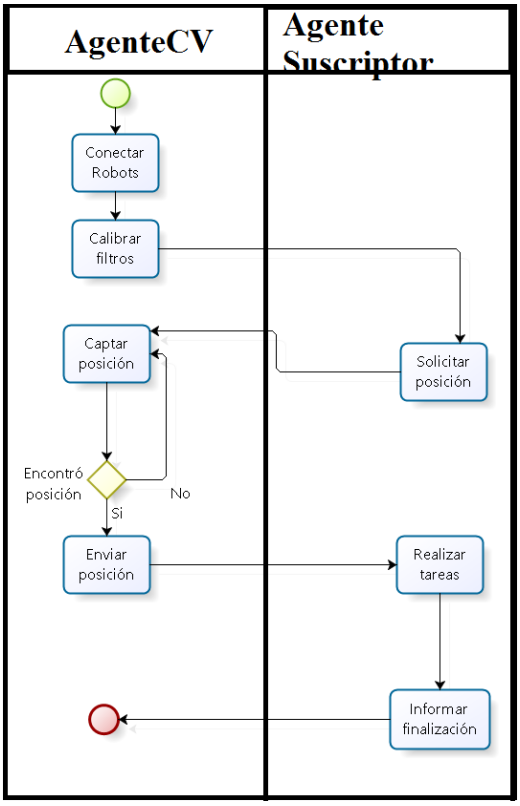

Figura 3.5 Especificación de Tareas

3.2.2 Modelo de sociedad de agentes

En estos modelos podemos ver la descripción de los agentes que se encuentran en el sistema en términos de roles, interacciones sociales, dependencias y ontologías.

3.2.2.1 Descripción de ontología de dominio

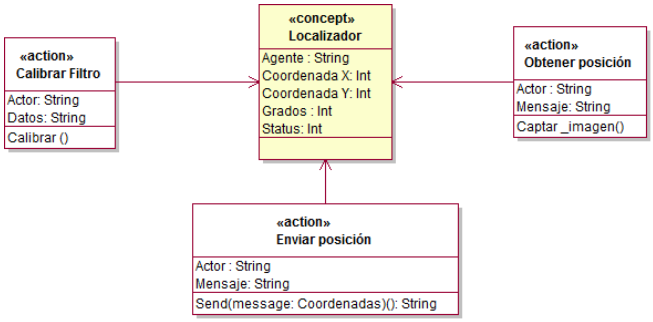

Figura 3.6 Descripción de Ontología del Dominio

El diagrama de la Figura 3.6 muestra la relación entre el concepto y las acciones que posee el sistema. En este diagrama se puede ver que el sistema al cual se le implemente el sistema de posicionamiento y direccionamiento, va a depender en gran parte del concepto Localizador.

3.2.2.2 Descripción de ontología de comunicación

En este modelo de ontología de comunicación los agentes se describen en término de su conocimiento. Por su parte la comunicación es representada por la relación que se encuentra entre los agentes que se están comunicando. En el siguiente diagrama se puede observar la relación que existe entre el AgenteCV y el Agente suscriptor, también se puede observar el tipo de comunicación que poseen para relacionarse.

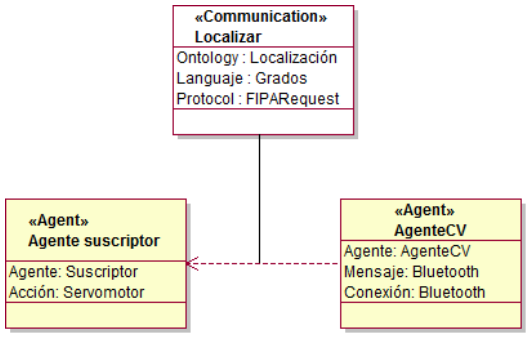

Figura 3.7 Descripción de ontología de Comunicación.

3.2.2.3 Descripción de roles

Esta descripción se representa por medio de un diagrama de clases, donde los roles que cumple cada uno de los agentes es una clase. Estos roles se comunican entre sí para así poder llegar a cumplir el fin que tiene el sistema multi-agente. Estos roles a su vez se componen de tareas con las cuales se logra conseguir llevar a cabo el rol que se le asigno. Para el actual sistema los roles que se desean satisfacer son las Tareas del suscriptor, las cuales dependen del sistema a cual se una el sistema de posicionamiento y la más importante obtener la posición en cuales se encuentran los NXT por medio de OpenCV.

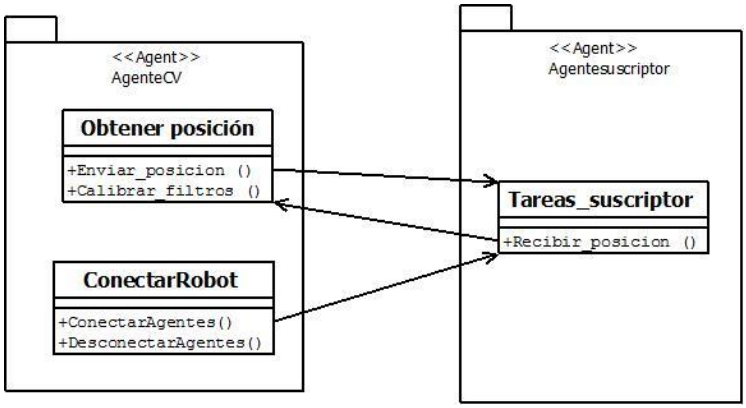

Figura 3.8 Descripción de Roles.

3.2.2.4 Descripción de Protocolos

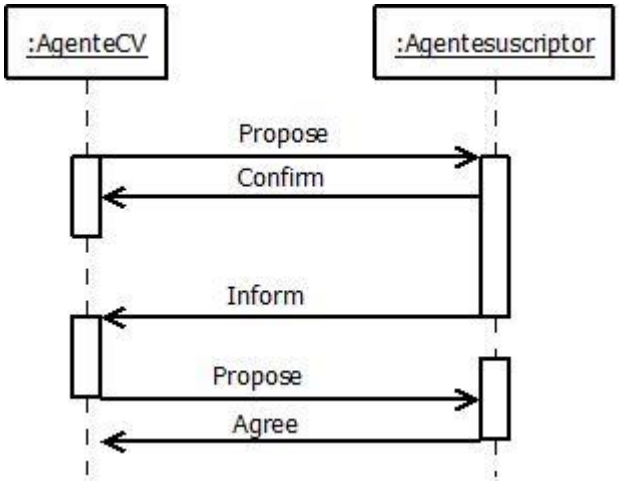

Figura 3.9 Descripción de Protocolos

La descripción de protocolos se representa por medio de un diagrama de secuencia. En este se representan las interacciones que ocurren entre los diferentes agentes que participan en el sistema. En la Figura 3.9 se puede observar como el agenteCV interactúa con el Agentesuscriptor, le envía la información necesaria como una propuesta, donde el suscriptor confirma su recepción. Terminado esto se le informa que ya la proceso y necesita más información.

3.2.3 Modelo Implementación de Agentes

En esta sección se describe la estructura que tiene el sistema multi-agente, tanto individualmente como en conjunto.

3.2.3.1 Definición de la estructura Multi-Agente

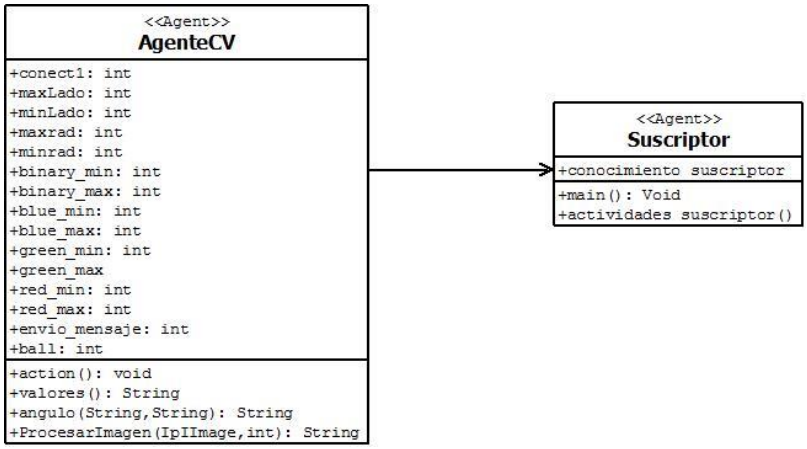

Figura 3.10 Definición de la estructura Multi-Agente

Este es un diagrama de clases donde las clases representan a los agentes que pertenecen al sistema. Los atributos de las clases representan el conocimiento que posee cada uno de los agentes dentro del sistema, mientras que las funciones representan las actividades que debe realizar este para llevar a cabo su finalidad dentro de esta relación. En el diagrama se pueden observar los agentes que actúan es este sistema, junto con el conocimiento que posee cada uno de ellos y sus tareas.

3.2.3.2 Definición de la estructura del Agente Individual

Esta etapa consiste en diseñar diferentes diagramas de clases para cada uno de los agentes involucrados en el sistema. Donde se puede observar la estructura del agente y de todas las tareas relacionadas a este. Para esto se especifican las clases propias de la plataforma utilizadas por el agente para funcionar.

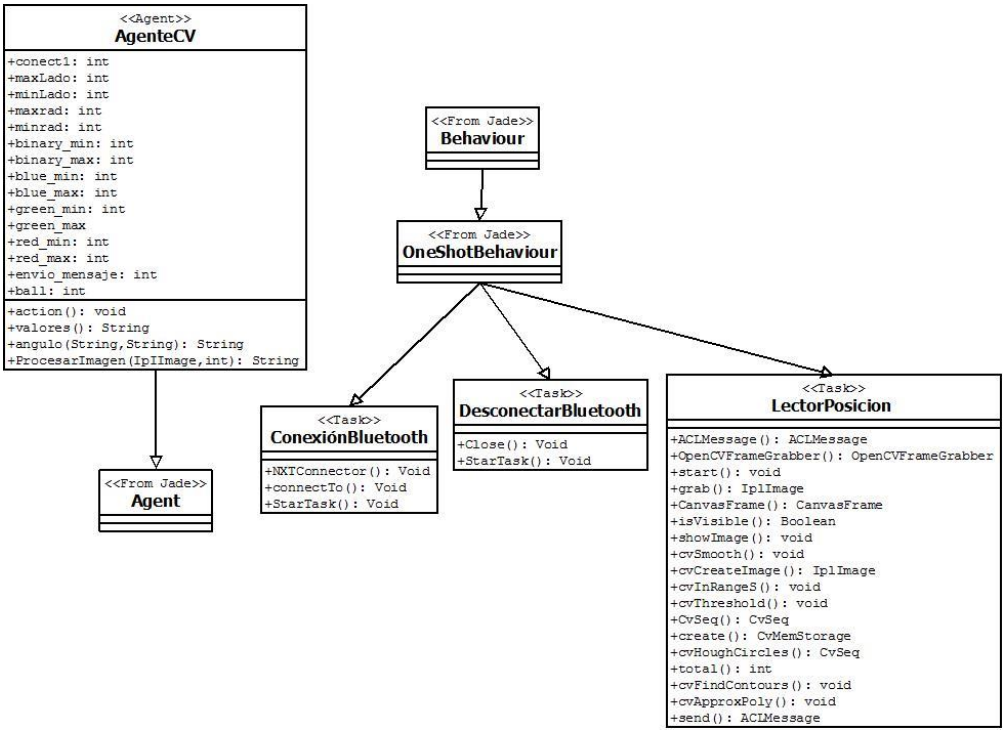

Figura 3.11 Definición de la estructura del Agente Individual, AgenteCV

El diagrama muestra la estructura individual del agenteCV que tiene la función de enviar la posición y direccionamiento en cual se encuentran actualmente los NXT, sus tareas extienden de la clase OneShotBehaviour, su principal tarea es localizar los robots y crear un mensaje que pueda ser comprendido por otros agentes que utilice el sistema.

3.2.4 Modelo de Despliegue

3.2.4.1 Configuración de despliegue

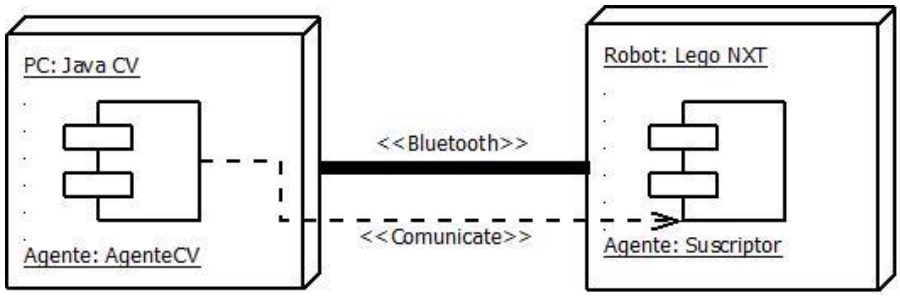

Figura 3.12 Modelo de Despliegue

El diagrama de despliegue se encarga de describir como los agentes se encuentran dispersos a través de las plataformas disponibles y de sus movimientos. Las plataformas se describen como nodos de proceso, los agentes como componentes y sus comunicaciones con líneas punteadas.

La Figura 3.12 representa un sistema en cual se tiene un solo NXT y el sistema de posicionamiento, el cual trabaja en dos plataformas: un computador y un lego NXT. Además se puede observar el tipo de comunicación existente entre las plataformas.

3.3 Desarrollo del Sistema de reconocimiento de posicionamiento y direccionamiento de lego robot a través de OpenCV

3.3.1 Procedimiento previo a la implementación

Para poder comenzar con la implementación del sistema de reconocimiento de posicionamiento y direccionamiento de lego robot a través de OpenCV, se debió configurar las herramientas con las que se va a trabajar.

- JAVA JDK: Lo primero que se debió realizar fue la instalación del JDK de java, esto es necesario si se va a trabajar con el lenguaje de programación Java, ya que este provee herramienta de desarrollo para la creación de programas Java. Este JDK se encuentra disponible en la siguiente página:
  - http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html

- Librerías JavaCV: En esta librería se encuentran todas las funciones que se utilizaran para la captura, análisis y procesamiento de la imagen. Estas se puede encontrar en la siguiente página (en este proyecto se utilizó la versión Java CV 0.10):
  - https://github.com/bytedeco/javacv

- Entorno de desarrollo Eclipse: Eclipse es un entorno de desarrollo para aplicación en java. Se escogió este entorno porque es un entorno familiar y utilizado en el transcurso de la carrera del estudiante. Este puede ser descargado de la siguiente página:
  - http://www.eclipse.org/downloads/

- Driver USB de Lego NXT: Este driver permite que el computador en cual se va a desarrollar el proyecto reconozca los Lego Mindstorms NXT al momento de conectarlos. Este puede ser descargado de la siguiente página:
  - http://mindstorms.lego.com/en-us/support/files/default.aspx#Driver

- LeJOS NXJ: Es un api, la cual permite reemplazar el firmware del ladrillo del NXT, para poder trabajar con el lenguaje Java. Además de esto ofrece una biblioteca de clases de Java, para permitir la programación de los Ladrillos.

También dispone de una serie de herramientas útiles para el uso de un NXT. Esto se puede descargar de la página:

- http://sourceforge.net/projects/lejos/files/lejos-NXJ-win32/

- Plataforma JADE: Es un plataforma la cual permite trabajar con agentes, desarrollada en el lenguaje de programación Java. Ofrece un biblioteca de clases en Java para el manejo de agente. También posee una interfaz de administración, con la cual se pueden realizar acciones, como consultar los agentes en ejecución, lanzar nuevos agentes, consultar sus estados, hacer debug, enviar mensajes ACL, entre otras. Esta plataforma se encuentra en la página:
  - http://jade.tilab.com

Una vez terminada la configuración de lo anteriormente mencionado, se debe configurar Eclipse para que pueda trabajar con LeJOS. Para esto se debe instalar el plugin de LeJOS, para poder hacerlo se debe ir a Eclipse-> Help-> Install New Software -> [agregar el link de LeJOS] ->Aceptar. El link que se debe agregar es "http://www.lejos.org/tools/eclipse/plugin/nxj/".

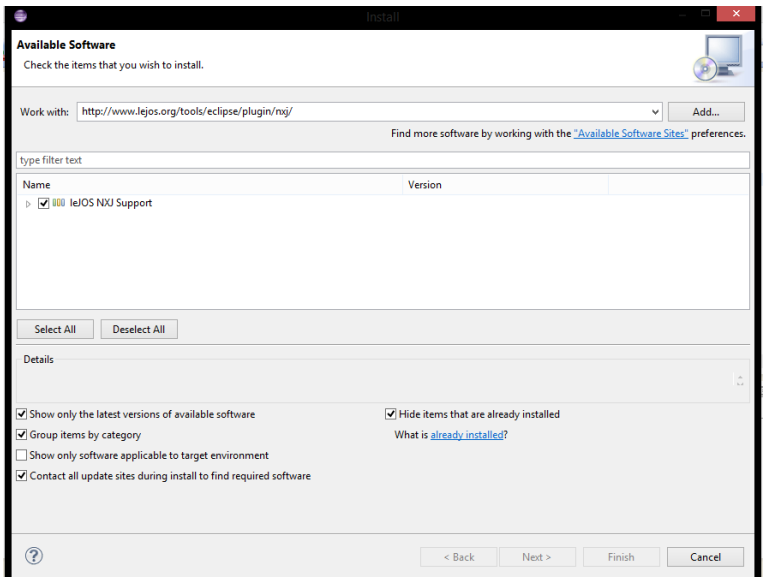

Figura 3.13 Instalación de LeJOS

Una vez terminado esto se debe crear un proyecto LeJOS PC e importar las librerías de JavaCV y Jade. Esto se realiza de la siguiente manera, primero descomprimir las librerías descargadas, luego dentro del proyecto en la barra superior seleccionar Project > Properties > Java Build Path > Libraries y seleccionar "Add External JARs". Luego de hacer la selección se deben localizar las librerías y seleccionarlas y para terminar presionar OK. En este proyecto las librerías que son esenciales de seleccionar son javacpp.jar, javacv.jar y Jade.jar.

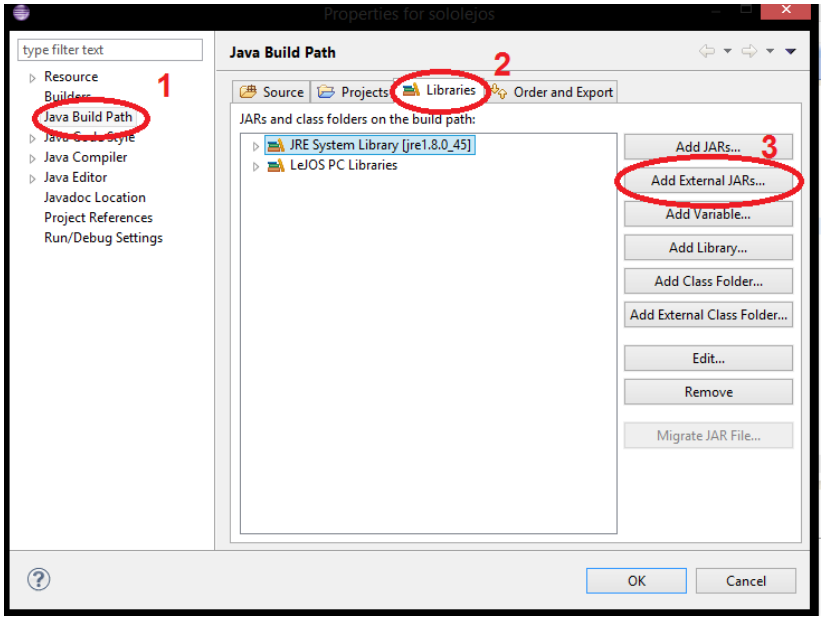

Figura 3.14 importación librerías

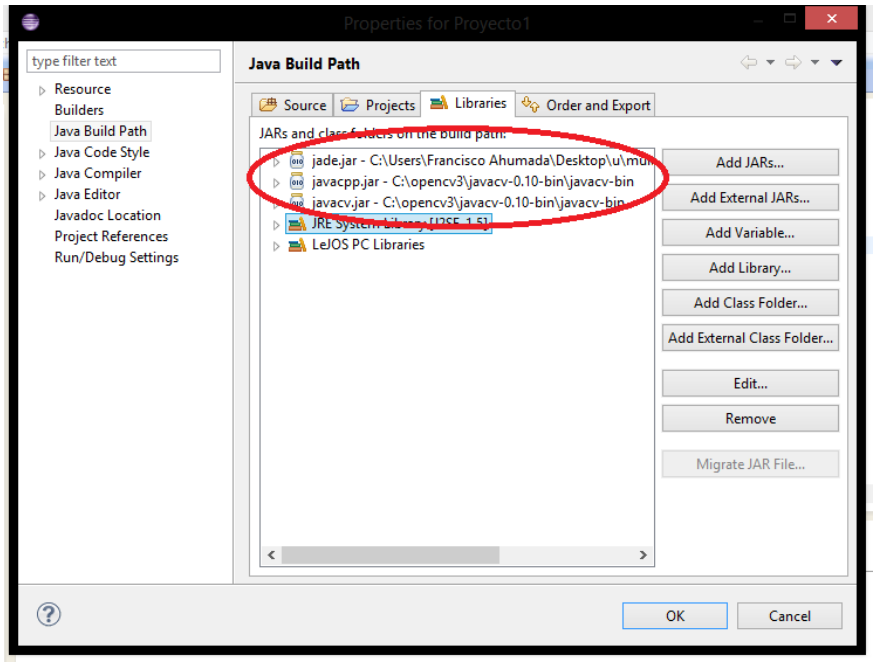

Figura 3.15 Resultado importación librerías

3.3.2 Implementación

En esta entrega se desarrolló un sistema multi-agente el cual es capaz detectar la posición y dirección de un Legos NXT, dentro de un escenario. Lo primero que se tuvo que realizar fue una interfaz, la cual permitiera auto ajustar las variables necesarias para realizar los filtros de imagen. En la Figura 3.16 se puede apreciar la versión final de la interfaz del proyecto.

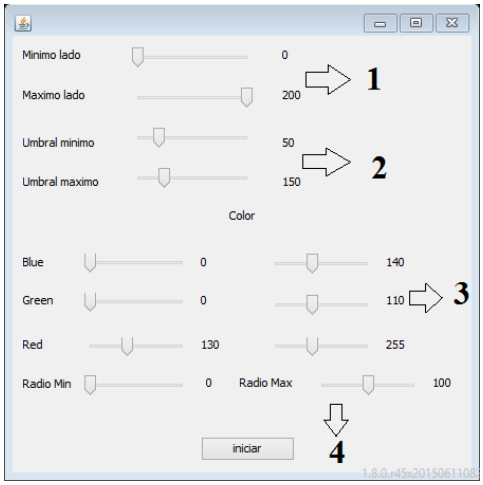

Figura 3.16 Interfaz del Sistema

1. Sector donde se ajusta el tamaño del triángulo que se quiere detectar, con un mínimo de 0 pixeles y un máximo de 200 pixeles. Este ajuste se realiza debido a que dependiendo de la altura el tamaño que se debe captar de la figura es diferente.
2. Sector donde se ajusta el umbral, con un mínimo de umbral 0 y un máximo de umbral 255. Este ajuste es para revisar cual umbral es mejor dependiendo de las características del espacio donde se está usando el sistema.
3. Sector donde se ajusta el color de las figuras que se desea detectar, el sistema trabaja con el sistema BGR, donde se tiene que poner el rango del color con cual se quiere trabajar.
4. Sector donde se detecta el tamaño de la circunferencia, esta sección se realizó pensando en algún futuro sistema que utilice el sistema que se desarrolló, que utilice algún tipo de balón.

Una vez ya realizada la interfaz, se procedió a codificar el sistema. Como primer paso se crearon los dos agentes que se van a utilizar. El primero de estos fue un agente suscriptor, este agente se encarga de simular posibles agentes final que pueden utilizar el sistema. Más adelante en la sección de casos de estudio se especificaran las tareas que tiene el agente en cada prueba. Como último y más importante se encuentra el AgenteCV, el que es el encargado de coordinar a los robots y de detectar la posición en cual se encuentran.

Al finalizar la creación de los agentes se les debe importar las librerías necesarias para el funcionamiento de estos.

Al AgenteCV se le incorporó las librerías necesarias para poder capturar, procesar y analizar imágenes y las librerías para trabajar con agentes, las cuales se importan de la siguiente manera:

```
import static org.bytedeco.javacpp.helper.opencv_core.CV_RGB;
import static org.bytedeco.javacpp.opencv_core.*
import static org.bytedeco.javacpp.opencv_imgproc.*;
import jade.core.Agent;
import jade.core.behaviours.*;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.util.StringTokenizer;
import org.bytedeco.javacpp.Loader;
import org.bytedeco.javacpp.opencv_core.*
import org.bytedeco.javacv.*
import lejos.nxt.Motor;
import lejos.pc.comm.NXTCommLogListener;
import lejos.pc.comm.NXTConnector;
```

Una vez creado el AgenteCV se le incorporo el funcionamiento del sistema de reconocimiento de triángulos que se entregó en Proyecto1. Ya con estas funcionalidades se procedió a incorporar la interfaz con el sistema existente. Lo primero que se incorporo fue la selección de dimensiones del triángulo, con el fin de poder tener el tamaño justo de este y que no se detectaran otros triángulos que se puedan generar por la sombra o algún objeto en el escenario. Para esto se realizó un if donde se comprueba el tamaño y si es de las medidas deseadas se realiza las tareas, como se muestra a continuación donde minLado es el tamaño mínimo que debe tener los lados del triángulo y maxLado es el tamaño máximo que debe tener los lados del triángulo.

```
if(lado1<=maxLado && lado2<=maxLado && lado3<=maxLado &&
lado1>=minLado && lado2>=minLado && lado3>=minLado){
        …procesamiento de datos…
}
```

El siguiente dato se incorporó por auto ajuste fue el umbral que se desea utilizar en el sistema, para esto se crearon las variables binary_min y binary_max, donde representan el

mínimo de umbral y el máximo de umbral que debe tener la imagen. A continuación se muestra como quedo en el código la selección del umbral.

```
cvThreshold(imgGrayScale,imgGrayScale,binary_min,binary_max,CV_THRESH_BI
NARY_INV);
```

Luego de esto se auto ajusto las variables de los colores, teniendo que crear seis variables las cuales representan los valores máximo y mínimo del rango del color a captar. Con esto se modificó la función cvInRangeS para poder incorporar estas variables y poder revisar vía pantalla como se detecta el color.

```
cvInRangeS(image, cvScalar(blue_min, green_min, red_min, 0), cvScalar(blue_max,
green_max, red_max, 0), imgGrayScale);
```

Como último para sacarle el mayor provecho al sistema, se incorporó una nueva función al sistema de posicionamiento el cual consiste en además de captar figuras de tres lados, fuera capaz de detectar circunferencias, pensando en futuros sistemas que puedan utilizar este sistema que utilicen algún tipo de balón. Para esto se tuvo que realizar un estudio respecto a cómo captar circunferencias por medio de OpenCV. Una vez realizado el estudio se procedio a desarrollar la nueva función para esto se utilizó cvHoughCircles, función la cual detecta circunferencias dentro de una imagen.

```
CvSeq circles = cvHoughCircles,imgGrayScale,storageBall,
CV_HOUGH_GRADIENT, 2, 100, 50, 50, 10, 500);
```

Para la detección de la circunferencias también se creó un sección en la interfaz para seleccionar el radio a cual se desea detectar, al igual que en la detección de triangulo se creó un if el cual indica en qué momento se debe procesar la imagen.

```
if(curCircle.z()<=maxrad && curCircle.z()>=minrad){
        …Procesamiento de datos…
}
```

Luego de realizar esto se observó que el sistema comenzaba a funcionar antes de terminar de ajustar los datos que se deseaban utilizar. Por lo que se tuvo que crear un variable la cual indica si los ajustes ya fueron realizados o no. Esta variable funciona de la siguiente manera, si tiene un valor 1 no envía mensajes y si aprieta el botón iniciar esta variable cambia a 0 indicando que debe enviar los mensajes.

3.4 Casos de estudio y pruebas

Las pruebas se realizan con el fin de poder detectar los errores y proporcionar información objetiva e independiente sobre la calidad del sistema. Por esta razón en el desarrollo de este avance se realizaron diferentes tipos de prueba a medida que se desarrollaba el sistema.

3.4.1 Entorno de Prueba

El entorno de prueba que fue usado para probar el sistema consiste en:

- Cancha de 87 cm de ancho y 65 cm de largo.
- Cámara fujitel HD de 720p de resolución a una altura de 1,72 metros.
- Balón azul.
- Lego Mindstorms NXT.

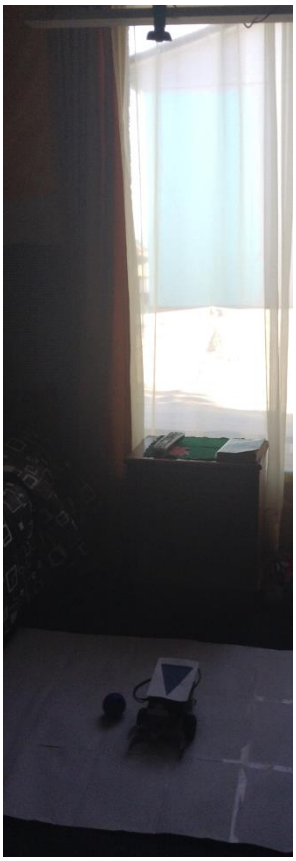

Figura 3.17 Entorno de Prueba

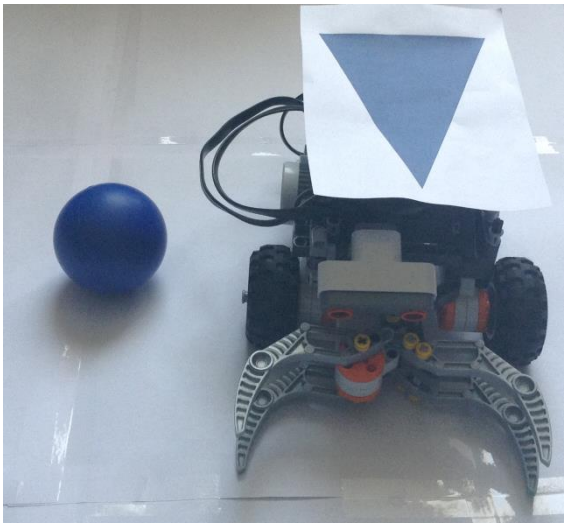

Figura 3.18 Lego Mindstorms NXT de prueba y Balón

3.4.2 Caso de estudio giro del NXT

Este caso de estudio tiene como finalidad determinar la precisión que se puede tener en los giros un NXT. Para esto el agente suscriptor simulo un sistema donde el NXT realiza un giro hasta el ángulo que se le indica. Para realizar las pruebas por medio de la interfaz se ingresaba el ángulo a cual se deseaba llegar. También una vez llegado al ángulo se giraba el NXT para ver si otra vez llegaba a la posición.

Los primeros resultados no fueron los deseados, esto debido a que al momento de llegar al ángulo deseado el NXT se detenía, pero se producía un pequeño derrape produciendo que el ángulo al momento de estar detenido no fuera el que se deseaba llegar. Este problema producía que el NXT volviera a girar y producir el mismo derrape otra vez y nunca detenerse. Para solucionar esto, se creó un factor de tolerancia el cual permite que el robot se detenga en un ángulo dentro del factor, por ejemplo si se tiene un factor de 2 y se desea llegar a 90, el NXT podrá detenerse entre el rango de 88 y 92, habiendo que el ángulo nuevo sea menos el factor y más el factor.

Al aplicar el factor se pudo apreciar que el sistema realiza bien los giros, pero también se concluyó que a mayor velocidad mayor tiene que ser el factor. A una velocidad de 100 grados por segundos un factor de 4 era suficiente para que se detuviera en la posición deseada, mientras que a una velocidad de 200 grados por segundos este factor era muy poco, por lo que se tuvo que probar con factores de mayor amplitud para que generada los resultados deseados.

3.4.3 Caso de estudio de Posicionamiento

Este caso de estudio tiene como finalidad poder medir la exactitud de llegada a un punto dado de un NXT. Para esto el agente suscriptor simulo un sistema donde dadas una coordenada x,y el NXT llega a dicha posición. Para realizar las pruebas de este caso, se enviaban por medio de la interfaz las coordenadas a cual se quería hacer llegar el NXT. Una vez que se llegaba a cierto punto se tomaba el NXT y se colocaba en otra posición para que volviera a llegar a la misma posición.

Los primeros resultados fueron los deseados, debido a que se ingresaba una coordenada y el NXT llegaba efectivamente al punto deseado. Las primeras pruebas fueron realizadas con un sistema que hacía que el NXT caminara en forma de L para llegar a la posición dada, como indica la figura.

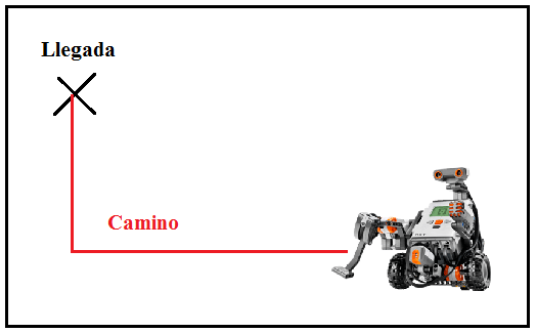

Figura 3.19 Diagrama camino en L

Al ver que estas pruebas fueron exitosas, se procedió a crear las mismas pruebas pero esta vez el NXT se dirigiría a la posición directamente y no por medio de un camino en forma de L. estas pruebas también fueron exitosas, debido a que se llegó a la posición deseada, la única complicación fue encontrar el ángulo en cual se debía dirigir el NXT.

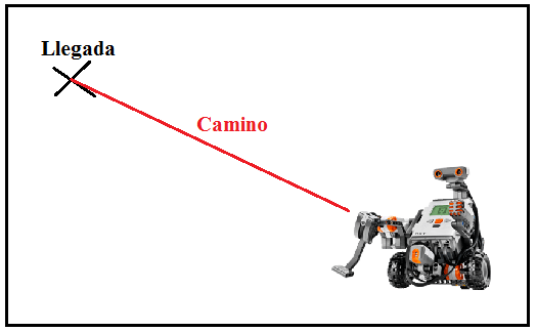

Figura 3.20 Diagrama camino directo

Con estas pruebas se pudo concluir que al igual que el giro, se necesita un factor para que el NXT se detenga, debido a que según el armado del robot el triángulo que identifica la posición puede que no esté en el centro y puede provocar que este ande más de lo que se necesita. Por lo tanto este factor va a depender específicamente del ensamblado del robot y la posición donde se le coloque el triángulo.

3.4.4 Caso de estudio tomar balón

El objetivo de este caso de estudio es poder probar la nueva funcionalidad del sistema, la cual consiste en detectar circunferencias por medio de OpenCV. Para esto el agente suscriptor simulo un sistema el cual solicita la posición del NXT y del balón. Para realizar las pruebas se utilizó un balón de color azul, igual que el triángulo del NXT.

Las primeras pruebas fueron medianamente satisfactorias, debido a que se lograba tener la posición del balón y el NXT llegaba a dicha posición pero no lograba obtener la pelota, esto debido a la forma en cómo se encontraba armado el NXT (problema hallado en caso de estudio de posicionamiento). Para poder solucionarlo se tuvo que hacer unos cambios en el NXT produciendo que se lograda tomar el balón.

Lo que se puede concluir de este caso es que el sistema de detección de circunferencia funciona, aunque se puede pulir para tener una mejor detección de circunferencias. También se puede demostrar que es sistema que se desarrollo puede ser implementados en sistemas que realicen tares diferentes a simplemente llegar de un punto a otro.

3.4.5 Caso de estudio pruebas de latencia

El objetivo de este caso de estudio es poder medir la latencia que se tiene al momento que el AgenteCV envía un mensaje y la recepción de este por el Agente suscriptor. Para esto se utilizó el agente suscriptor del caso de estudio de posicionamiento. Con lo cual se tomaron tres muestras diferentes, la muestra uno cuenta con un tamaño de 30 tiempos y las muestras dos y tres cuentan con 100 tiempos (en el Anexo D se pueden apreciar la tabla de tiempos). La muestra uno fue de menos tiempo solo para ver la variación de los datos en una muestra menor.

Por los resultados obtenidos por las muestra se puede apreciar que el promedio de latencia entre el envío y recepción del mensaje es aproximadamente de 1 milisegundo, con una tendencia a variar de aproximadamente 0,5 milisegundo. Tiempos que son muy insignificantes para la percepción humana pero para el sistema pueden producir error en el funcionamiento si no se tiene presente.

4 Conclusión

La robótica es un campo más complejo de lo que puede imaginarse. Un ejemplo de este campo es la visión artificial. Esta no solo es que el computador tome una imagen por algún tipo de cámara y arroje resultados. Lo que realmente ocurre es, una vez tomada la imagen el computador la digitaliza para así poder empezar a procesarla por medio de los diferentes tipos de filtros, que al transcurso de los años se han creado con el fin de poder simular la visión humana.

También en esta área se encuentra la inteligencia artificial, en la cual se intenta simular lo mejor posible el comportamiento humano por medio de un robot. Este es una de las áreas que más se relaciona con las demás áreas. Esto debido que con las demás áreas involucradas se puede recrear de mayor manera las acciones humanas.

En esta entrega final se implementó el sistema de posicionamiento y direccionamiento de lego robot por medio de OpenCV, con el cual se pretende simular una visión artificial en un Lego NXT, por medio de JavaCV y LeJOS. Esto produjo algunas complicaciones debido al conocimiento previo del proyectista sobre LeJOS. Una vez comprendido de una mayor manera este lenguaje, se pudo concluir que la recreación de un escenario no es cosa de un segundo a otro. Debido a que el robot tiene que tener un gran conocimiento del medio en cual se encuentra. Para esto se desarrollaron agentes los cuales contienen la información necesaria que se debe saber sobre el medio en cual se encuentra.

Los resultados que se deseaban obtener fueron satisfactorios, debido a que se logró desarrollar el sistema de posicionamiento y direccionamiento de lego robot por medio de OpenCV, con esto se pudo concluir que es posible darle una simulación de visión artificial a los NXT. Estas conclusiones fueron sacaras debido a las pruebas realizadas en las cuatro entregas del proyecto, donde se realizaron diferentes tipos de pruebas y utilizando los datos de diferente manera.

Para finalizar, también queda como trabajo futuro mejorar el sistema de detección de circunferencias. Con el fin de poder tener una mejor detección de circunferencias en el sistema, debido que solo se realizaron pruebas con un solo balón. También quedo como trabajo futuro la incorporación de este sistema en un sistema donde trabajen múltiples NXT y que necesiten coordenadas más exactas para realizar sus tareas.

5 Referencias

[1] Laboratorio Nacional de Calidad del Software, España, INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS DE VIDA 2009, pág. 24 - 38.

[2] Definición palabra visión, Extraída desde el diccionario de la RAE, disponible en la web: http://lema.rae.es/drae/srv/search?id=w2nMxD7W0DXX2SjulzP3, visitado la última vez el 13 de enero del 2016

[3] J. Sanabria y J. Archila, DETECCIÓN Y ANÁLISIS DE MOVIMIENTO USANDO VISIÓN ARTIFICIAL, Scientia et Technica Año XVI, No 49, Diciembre de 2011. Universidad Tecnológica de Pereira.

[4] Grupo de Investigación EDMANS, TÉCNICAS Y ALGORITMOS BÁSICOS DE VISIÓN ARTIFICIAL, 2006, pág. 33 - 72.

[5] J. Vélez, A. Moreno, A. Sánchez, J. Sánchez-Marín, Visión Por Computador, 2003, pág. 13-165.

[6] Universidad Nacional de Quilmes, Filtrado Espacial, 2005, documento de cátedra.

[7] Definición palabra convolución, Extraído del Consejo Superior de Investigaciones Científicas, disponible en la web: http://www.ija.csic.es/gt/tele/TUTORIAL%20A.I/definitions/convolucion.htm, visitado por última vez el 13 de enero del 2016

[8] Sitio Web oficial de la librería OpenCV: http://opencv.org/, Visitado por última vez el 13 de enero del 2016

[9] Enciclopedia libre: http://es.wikipedia.org/wiki/OpenCV, Visitado por última vez el 13 de enero del 2016

[10] S. Bustamante, Algoritmo de procesamiento de imagen aplicados a la deteccion de figuras geometricas y sus propiedades espaciales , PUCV, 2014, Memoria de Titulo.

[11] A. Catalán, Desarrollo de un Sistema Multi-Agente Colaborativo para Soccer-Robot, PUCV, 2013, Memoria de Titulo.

[12] L. Sucar, G. Gómez, Visión Computacional, Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla, México, Pág.1 - 49.

[13] V. Botti, A. Giret, Aplicaciones Industriales de los Sistemas Multiagente, Universidad Politécnica de Valencia.

[14] S. Russel and P. Norvig. “Artificial Intelligence: A Modern Approach”. Prentice Hall, 2nd edición, 2003, Pág. 3 - 31.

[15] Wooldridge, M. “Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence”. Ed. G. Weiss, MIT Press, Massachusetts, USA. 1999, pág. 27 - 118.

[16] Enciclopedia libre https://es.wikipedia.org/wiki/Java_Agent_Development_Framework, Visitado por última vez el 13 de enero del 2016

[17] Patricia Moreno Buisson, Análisis, Desarrollo e Implementación de un Mercado Virtual de Libros en la Web, Escuela Politécnica Superior de Jaén, Memoria de Proyecto.

[18] Martí Navarro Llácer, Gestión de Compromisos en Sistemas Multi-Agente de Tiempo Real, Valencia, España, 2011, Tesis Doctoral.

[19] Claudio Cubillos, Tecnología Multiagente: AOSE Escuela de Ingeniería Informática – PUCV, 2007, Documento de cátedra.

[20] Gustavo Acosta Amaya, AMBIENTE MULTI-AGENTE ROBÓTICO PARA LA NAVEGACIÓN COLABORATIVA EN ESCENARIOS ESTRUCTURADOS, Universidad Nacional de Colombia, 2010, Tesis Magister.

[21] Sitio Web oficial de Lego Mindstorms http://legomindstorms.es/, Visitado por última vez el 13 de enero del 2016

[22] Adrián Cervera Andes, Coordinación y control de robots móviles basado en agentes, Valencia, España, 2011, Tesis.

Anexos

A. Desarrollo del Sistema de detección de triángulos

A.1 Implementación

En esta entrega de proyecto se desarrolló un sistema el cual es capaz de reconocer un triángulo a través de la cámara del computador. Se escogió como figura geométrica el triángulo, esto debido a las ventajas que se tiene a usar esta figura al momento de detectar la posición y dirección de este. Otra característica para la toma de esta decisión es la cantidad de vértices, esto produce que sea más fácil su detección.

Lo primero que se realizo fue importar las librerías necesarias para poder capturar, procesar y analizar imágenes, las cuales se importan de la siguiente manera:

```
import org.bytedeco.javacpp.Loader;

import org.bytedeco.javacpp.opencv_core.IplImage;

import org.bytedeco.javacv.CanvasFrame;

import org.bytedeco.javacv.OpenCVFrameGrabber;

import static org.bytedeco.javacpp.opencv_imgproc.*;

import static org.bytedeco.javacpp.opencv_core.*;

import static org.bytedeco.javacpp.opencv_highgui.*;
```

Luego de tener las librerías importadas se procede a la captura de video, esto se realiza a través del siguiente procedimiento, primero se ocupan las funciones OpenCVFrameGrabber() y grabber.start() cuales tienen como función la lectura de un flujo de vídeo de la cámara. Una vez terminado esto se almacena el fotograma del video, grab() es el encargado de realizar esta tarea. Como último paso de la captura de video se crea la venta en cual se podrán observar los resultados del procesamiento de la imagen con la función CanvasFrame().

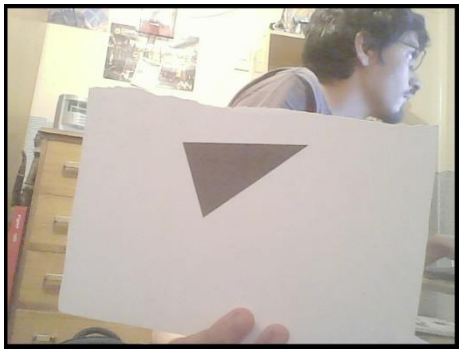

Figura A.1 Captura de pantalla

El siguiente paso se le aplican diferentes procesos a la imagen, para facilitar la detección de los triángulos. Primero se le aplica un filtrado de suavidad Gaussiano cvSmooth(). Luego se procede a cambiar la imagen a escala de grises con la función cvCreateImage(). Como último se selecciona el umbral que se va a utilizar, para así poder eliminar el ruido de la imagen y para obtener la imagen en niveles binarios.

Figura A.2 Imagen al aplicar los filtros

La terminar de realizar todos los filtrados de la imagen, se utiliza la función cvFindContours(), con la cual se obtienen todos los contornos de la imagen. Luego se analizan los contornos detectados, para revisar si la imagen tomada es un triángulo o no esto se realiza con cvApproxPoly(), con esta se verifica que la figura tenga tres lados. Luego se extraen los vértices de este para así poder obtener los valores que se desean saber de este. La primera información relevante que se obtiene de estos es la medida de sus lados en píxeles, se obtienen con la fórmula de la distancia entre dos puntos:

$$d = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

Donde (X_1,Y1) y (X_2,Y_2) son uno de los pares de vértices obtenidos. Una vez con los datos de cuanto mide cada lado se hacen dos operaciones la primera se revisa que todos los lados midan más de 40 píxeles, esto se realiza para evitar que el programa detecte los triángulos que se pueden generar en las superficies u objetos cercanos que no corresponden al ambiente del sistema. El segundo proceso es la elección del menor lado, el cual será

seleccionado como la base del triángulo, para este prototipo se tomó en cuenta que dos o más lados son el menor se toma uno de esos igual como base.

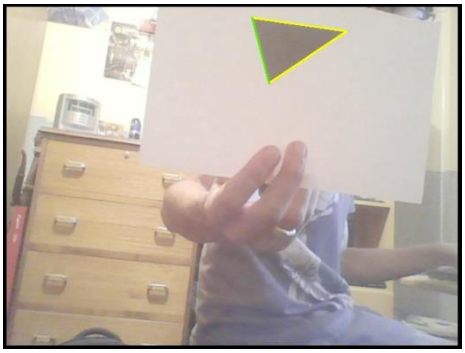

Figura A.3 Imagen detección de base

Lo siguiente a calcular es el centro del triángulo, para así saber en qué parte de la imagen se encuentra, la ecuación a usar es el promedio de las coordenadas x e y de los vértices:

$$X_{medio} = \frac{X_1 + X_2 + X_3}{3}$$

$$Y_{medio} = \frac{Y_1 + Y_2 + Y_3}{3}$$

Donde (X_1,Y1), (X_2,Y_2) y (X_3,Y_3) son los vértices del triángulo y (X_{medio},Y_{medio}) son el centro del triángulo. Al terminar esto se calcula el punto medio de la base similarmente al cálculo anterior solo que esta vez con los vértices que se encuentran en la base y se obtienen (X_{base},Y_{base}), este punto se utiliza para el cálculo de inclinación respecto al eje x del triángulo. La forma de cálculo de este dato consiste en crear un triángulo imaginario utilizando las coordenadas del vértice opuesto a la base y las coordenadas del punto medio de la base. Las formulas a utilizar son:

$$Angulo = \arccos(\frac{CA}{HI})$$

Donde:

$$CA = \sqrt{(X_{base} - X_{opuesto\ vertice})^2} \quad \text{Y}$$

$$HI = \sqrt{(X_{base} - X_{opuesto\ vertice})^2 + (Y_{base} - Y_{opuesto\ vertice})^2}$$

Al terminar este cálculo se muestra por pantalla la inclinación en grados del triángulo y las coordenadas de su centro.

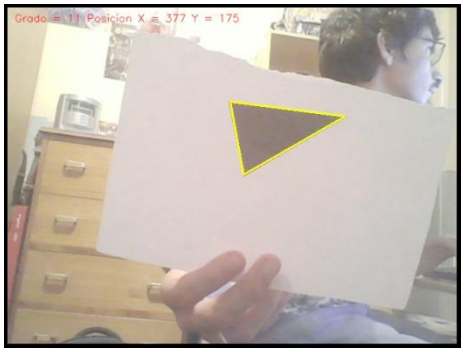

Figura A.4 Imagen resultado final

A.2 Pruebas

Las pruebas se realizan con el fin de poder detectar los errores y proporcionar información objetiva e independiente sobre la calidad del sistema. Por esta razón en el desarrollo de este avance se realizaron diferentes tipos de prueba a medida que se desarrollaba es sistema.

A.2.1 Prueba de detección de figura de una imagen

Esta prueba consistió en poder detectar un triángulo por medio de JavaCV. Para esta prueba se le ingreso al sistema una imagen predeterminada, para así poder aplicarle los filtros de imagen y así poder reconocer si existe o no un triángulo.

Los resultados obtenidos en esta prueba fueron satisfactorios, debido a que en internet se lograron encontrar ejemplos similares a lo que se deseaba implementar en esta fase pero en Lenguaje C++, lo que provoco que esta fase fuera la mayor parte traspasar de C++ a Java.

A.2.2 Prueba de detección de figura por cámara

Al igual que la prueba anterior el objetivo de esta prueba era la detección de un triángulo por medio de JavaCV, con la diferencia que esta vez se utilizaría la cámara para obtener la imagen a procesar. Para esta prueba se necesitó contar con una cámara conectada a la computadora para poder realizar las pruebas.

Los primeros resultados de esta prueba no fueron los esperados, se produjeron dos tipos de resultados. El primero fue exitoso, que fue el poder captar por la cámara del computador la imagen de donde se quería detectar la figura, esto debido a que en la investigación previa al inicio del proyecto que estudiante realizo estudio la forma de como captar las imágenes de la cámara por medio de JavaCV. El otro resultado no fue completamente exitoso, esto debido a que se pudo reconocer el triángulo que se mostraba a la cámara, pero a su vez reconoció otros triángulos que se encontraba en el ambiente de pruebas

(triángulos en las murallas, en la ropa, entre otros), lo que género que se tuviera que realizar mejoras al sistema.

Para solucionar los problemas en esta fase se tuvo que realizarse un análisis para saber qué tipos de triángulos se deseaba captar. Las dos opciones que se estudiaron fueron por el área del triángulo o por el tamaño de los lados de este. Se llegó a la decisión que se utilizaran los triángulos cuyos lados son mayor a 40 píxeles. Con esto se provocaron los resultados esperados reconociendo solo el triángulo que se deseaba captar.

A.2.3 Pruebas de obtención de información

El objetivo de esta prueba es poder tener toda la información relevante de la imagen, es decir la posición donde se encuentra el triángulo y su ángulo de inclinación. Para esta prueba era necesario ya tener el prototipo completamente terminado.

Los resultados de esta prueba fueron de dos tipo, satisfactorios y negativos. Los resultados satisfactorios se debieron a que se pudo detectar la posición del triángulo, esto fue porque al tener todos los vértices del triángulo solo se tuvo que programar una función matemática para calcular el centro de este. La negativa fue el cálculo del ángulo de inclinación, esto se produjo por la falta de conocimiento de la función acos() por parte del estudiante, la solución a esto se produjo investigando cómo funcionaba esta función.

Luego de tener la solución al problema se produjo otra falla, la cual consistía en calcular de mal manera el ángulo de inclinación. Esta falla se debía a que el eje Y se estaba tomando de mal manera, es decir se consideró que el eje vertical partía vertical mente de 0 hacia arriba, produciendo que los cálculos fueran errados, luego de una poco de análisis se encontró la solución la cual era partir el eje vertical desde 0 hacia abajo.

B. Desarrollo del Sistema Multi-Agente Robot-OpenCV

B.1 Implementación

En esta entrega de proyecto se desarrolló un sistema multi-agente el cual es capaz de reconocer la posición en cual se encuentra dentro de un escenario un Lego NXT. Para realizar esto se utilizó el prototipo entregado en el avance de proyecto 1, el que consistía en reconocer un triángulo a través de la cámara del computador.

Lo primero que se realizó fue crear los agentes con los que se trabajará, los cuales son un AgenteCV, el que tendrá incorporado el sistema de detección de triángulos y será el encargado de enviar constantemente la posición en cual se encuentra el NXT. El otro agente a utilizar es un AgenteRobot, el que será el encargado de recibir el mensaje enviado por el AgenteCV para poder decidir hacia donde mover el Lego.

Una vez creado ambos agentes se le importo las librerías necesarias a cada uno para poder funcionar.

Al AgenteCV se le incorporó las librerías necesarias para poder capturar, procesar y analizar imágenes y las librerías para trabajar con agentes, las cuales se importan de la siguiente manera:

```
import java.util.Scanner;

import org.bytedeco.javacpp.Loader;

import org.bytedeco.javacpp.opencv_core.IplImage;

import org.bytedeco.javacv.CanvasFrame;

import org.bytedeco.javacv.OpenCVFrameGrabber;

import static org.bytedeco.javacpp.opencv_imgproc.*;

import static org.bytedeco.javacpp.opencv_core.*;

import static org.bytedeco.javacpp.opencv_highgui.*;

import jade.core.*;

import jade.core.behaviours.*;

import jade.lang.acl.*;
```

Por otro lado al AgenteRobot se le incorporaron las librerías necesarias para poder hacer funcionar los Lego NXT y las librerías para trabajar con agentes, las que son las siguientes:

```
import lejos.nxt.Button;

import lejos.nxt.Motor;

import lejos.nxt.SensorPort;

import lejos.nxt.SoundSensor;

import lejos.nxt.TouchSensor;

import java.util.StringTokenizer;

import jade.core.*;

import jade.core.behaviours.*;

import jade.lang.acl.ACLMessage;
```

Luego de tener las librerías importadas se procede a la selección de los tipos de Behaviour que se van a utilizar en cada uno de los agentes. El AgenteCV se le selecciono un TickerBehaviour, esto debido a que debe enviar constantemente mensajes al AgenteRobot pero sin colapsarlo en mensajes, para este se escogió enviar mensajes cada 20 milisegundos para así tener un tiempo prudente por envió de mensaje, para esto se utilizó, addBehaviour(new EmisorComportaminento(this,20)). Mientras que el AgenteRobot se seleccionó un CyclicBehaviour, ya que este debe estar constantemente solicitando y recibiendo mensajes para poder saber en qué posición se encuentra, donde se ocupó addBehaviour(new ReceptorComportaminento()).

Una vez hecha la selección se procede a la creación del mensaje que el AgenteCV envía al AgenteRobot, la detección de triángulos se encuentra explicada en el Anexo A. Lo primero que se creo fue un ACLMessage del tipo Request de la siguiente manera:

```
ACLMessage mensaje = new ACLMessage(ACLMessage.REQUEST).
```

Una vez creado este se le setean los datos necesarios para poder crear un mensaje, antes de mostrar la forma en cual setear los datos se debe tener en cuenta que el mensaje se envía en forma de string de la siguiente manera “ángulo”-“posición x”-“posición y”. Realizar el mensaje de esta forma se eligió para poder integrar los tres datos de interés de la detección del triángulo en un solo mensaje. Continuando con la creación del mensaje, el procedimiento es el siguiente:

```
AID id = new AID();

id.setLocalName("agenterobot");

mensaje.setSender(getAID());
```

```
mensaje.setLanguage("Grados");

mensaje.addReceiver(id);

 datos=(int)angulo+"-"+mediox+"-"+medioy;

 mensaje.setContent(datos);

 send(mensaje);
```

Una vez el emisor envió el mensaje, el AgenteRobot crea un ACLMessage en cual contendrá la información que fue enviada por el AgenteCV, esto se realiza de la siguiente forma:

```
ACLMessage mensaje = receive();
```

Una vez realizado esto, el agente decide que realizar, si el mensaje que fue recibido se encuentra nulo no realiza ninguna acción, hasta que recibir un mensaje con algún contenido. Una vez llegado un mensaje con contenido valido se procede a la descomposición del mensaje, en palabras más entendibles a separar el mensaje en ángulo, posición x y posición y. para esto se utiliza StringTokenizer, que es parte de las librerías de Java.

```
h=mensaje.getContent();

StringTokenizer multiTokenizer = new StringTokenizer(h, "-");

g= Integer.parseInt(multiTokenizer.nextToken());

x= Integer.parseInt(multiTokenizer.nextToken());

y= Integer.parseInt(multiTokenizer.nextToken());
```

Una vez obtenidas las coordenadas se procede a verificar si se encuentran dentro de las dimensiones de la cancha. Si estas se encuentran dentro de estas dimensiones se verifica si es que llegaron al rango de giro o no, este rango es una cancha virtual de menor medida que la real en los ejes x e y. Las medidas a restar a la cancha real dependen de la altura en cual se encuentre la cámara. Esto se realiza para el momento que el robot llegue a esos límites se realice el giro e impida que se salga del campo asignado para desplazarse.

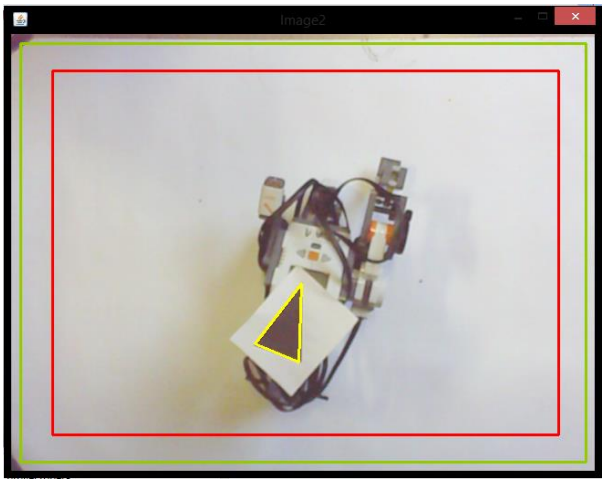

Figura B.1 Cancha Virtual del Sistema

Si el robot está dentro de la cancha virtual, éste debe avanzar o continuar avanzando dependiendo el caso en que se encuentre. Al momento del robot llegar a los límites de la cancha virtual este realiza un giro aproximado de 180°, y a su vez el auxiliar de giro cambia de valor de 0 a 1. Este auxiliar se ocupa para que el robot una vez que gire descarte cinco mensajes y así no dejar el robot girando infinitamente.

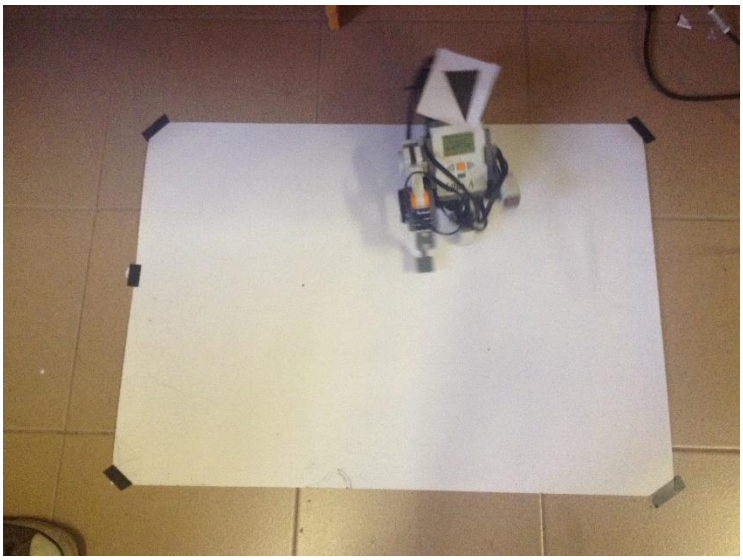

Figura B.2 Sistema Funcionando

B.2 Pruebas

Las pruebas se realizan con el fin de poder detectar los errores y proporcionar información objetiva e independiente sobre la calidad del sistema. Por esta razón en el desarrollo de este avance se realizaron diferentes tipos de prueba a medida que se desarrollaba el sistema.

B.2.1 Entorno de Prueba

El entorno de prueba que fue usado para probar el sistema consiste en:

- Cancha de 77 cm de ancho y 55 cm de largo.
- Cámara Genius VGA de 480p a una altura de 80 cm.
- Cancha Virtual de 65 cm de ancho y 43 cm de largo.
- Lego Mindstorms NXT.

Figura B.3 Entorno de Prueba

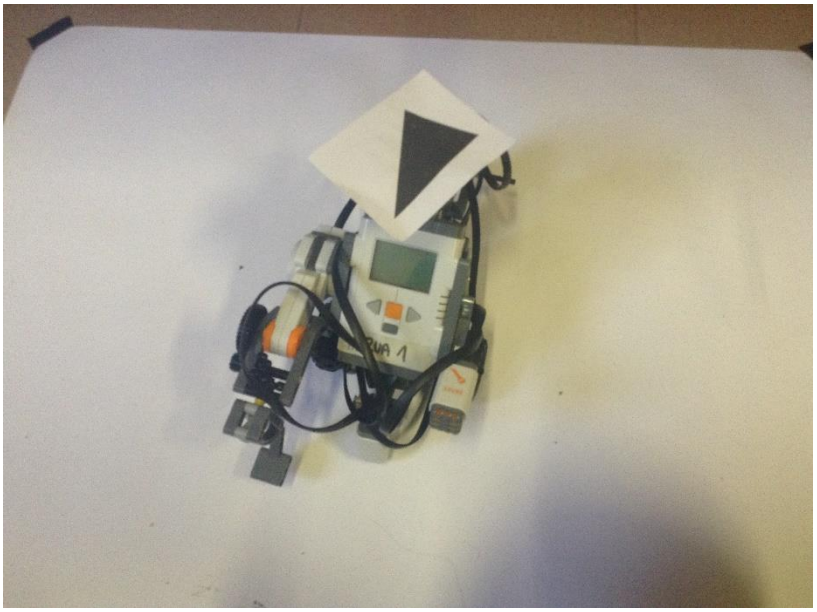

Figura B.4 Lego Mindstorms NXT de prueba

B.2.2 Prueba de comunicación entre agentes

Esta prueba consistió en poder revisar si los agentes se podían enviar los mensajes uno al otro. Para esto se creó un proyecto LeJOS PC, el cual consistía que al momento de recibir un mensaje el AgenteRobot este activara el servomotor que se encuentra en la mano del NXT. Para realizar esta prueba no se integró el sistema de detección de triángulos.

Los resultado es esta prueba fueron los deseados, debido a la amplia documentación de comunicación entre agentes en internet. Por esta razón no fue de gran complejidad lograr la comunicación entre el AgenteCV y AgenteRobot.

B.2.3 Prueba de Posicionamiento en el área de prueba

Ésta tenía como finalidad integrar el sistema de detección de triángulos, que se muestra en los Anexos al AgenteCV. Además revisar si el NXT se mantenía dentro de la cancha por medio de los mensajes enviados por el AgenteRobot. Lo primero que se probó fue que el sistema de detección envíe los mensajes con las posiciones en la que se encuentra el robot. La segunda prueba en esta fase fue que el mindstorms se detuviera exactamente al momento que llegara a los bordes de la cancha.

Los resultados obtenidos en la primera prueba de esta fueron medianamente éxitos, esto debido a que enviaba el mensaje pero no se logró enviar las tres coordenadas que se deseaban obtener. Para poder solucionar este inconveniente se creó un string en cual se almacenan las tres coordenadas separadas por un “-“, como por ejemplo “350-34-145”. Este

mensaje al ser recibido por el AgenteRobot se descompone dejándolo en tres variables diferentes.

Una vez solucionado el primer error, se probó que el robot se detuviera en las coordenadas del borde de la cancha. Para esta prueba los resultados no fueron los deseados, debido que el NXT no se detenía en la posición deseada. Esto ocurría, porque el triángulo no lograba llegar a los límites de la cancha. Para esto se creó una cancha virtual la cual es de 6 cm más pequeña por los cuatros costados de la cancha. Con esto los resultados obtenidos fueron los deseados para seguir avanzando en el proyecto.

B.2.4 Pruebas de Giro del NXT

El objetivo de esta prueba es hacer que el Lego pueda andar por dentro de la cancha sin salirse de esta. Para realizar esto se utilizaron los resultados obtenidos en la prueba de posicionamiento en área de prueba. Las diferencias con la prueba ya mencionada es que al momento de llegar las dimensiones de la cancha virtual este realice un giro de aproximadamente 180 grados.

Los resultados en esta prueba no fueron los esperados, debido a que al momento de realizar un giro el NXT permanecía constantemente girando. Esto ocurría debido a que el sistema no diferenciaba si se encontraba en movimiento o rotando. Para solucionar esto se creó un booleano el cual indica si el robot se encuentra girando o no. Si se encuentra en rotación se descartan los siguientes 5 mensajes recibidos. Con esto se logró conseguir los resultados que se deseaban obtener.

C. Desarrollo Sistema multi-agente para interacción de dos Legos NXT.

C.1 Implementación

En esta entrega de proyecto se desarrolló un sistema multi-agente el cual es capaz de permitir la interacción de dos Legos NXT. Lo primero que se tuvo que realizar fue la construcción de los dos NXT a utilizar, el resultado final de esto se puede ver en la Figura 3.19 y Figura 3.20.

Figura C.1 Lego NXT Proyecto costado

Figura C.2 Lego NXT Proyecto frente

Una vez ya ensamblados los robots, se procedió a codificar el sistema. Como primer paso se crearon los tres agentes que se van a utilizar. El primero de estos es el JugadorA, el que será el encargado de realizar la mayor parte de las tareas del sistema, entre las más importantes a cargo de este se encuentra la de dejar el balón en la posición adecuada para que el otro jugador tome este balón. Otro agente es el JugadorB, cual tiene como tarea tomar el balón una vez que el JugadorA lo suelte, para así terminar el funcionamiento del sistema. Como último y más importante se encuentra el AgenteCV, el que es el encargado de coordinar a los robots y de detectar la posición en cual se encuentran.

Al finalizar la creación de los agentes se les debe importar las librerías necesarias para el funcionamiento de estos.

Al AgenteCV se le incorporó las librerías necesarias para poder capturar, procesar y analizar imágenes y las librerías para trabajar con agentes, las cuales se importan de la siguiente manera:

```
import static org.bytedeco.javacpp.helper.opencv_core.CV_RGB;
import static org.bytedeco.javacpp.opencv_core.*
import static org.bytedeco.javacpp.opencv_imgproc.*;
import jade.core.Agent;
import jade.core.behaviours.*;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.util.StringTokenizer;
import org.bytedeco.javacpp.Loader;
import org.bytedeco.javacpp.opencv_core.*
import org.bytedeco.javacv.*
import lejos.nxt.Motor;
import lejos.pc.comm.NXTCommLogListener;
import lejos.pc.comm.NXTConnector;
```

Por otro lado a los Agentes Jugadores se les incorporaron las librerías necesarias para poder hacer funcionar los Lego NXT y las librerías para trabajar con agentes, las que son las siguientes:

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import lejos.nxt.Button;
import lejos.nxt.LCD;
import lejos.nxt.Motor;
import lejos.nxt.SensorPort;
import lejos.nxt.UltrasonicSensor;
import lejos.nxt.addon.CompassHTSensor;
import lejos.nxt.addon.IRSeekerV2;
import lejos.nxt.comm.BTConnection;
```

```
import lejos.nxt.comm.Bluetooth;
import lejos.robotics.navigation.DifferentialPilot;
import lejos.robotics.subsumption.Arbitrator;
import lejos.robotics.subsumption.Behavior;
```

Luego de tener todos los agentes con sus librerías correspondientes se procedió a desarrollar la función que permitiera a los Legos NXT tomar el balón. Para esto se trabajó con los sensores de distancia y el sensor IRSeeker V2.

```
seeker = new IRSeekerV2(SensorPort.S2, IRSeekerV2.Mode.AC);
sonic2 = new UltrasonicSensor(SensorPort.S1);
```

Antes de explicar cómo funciona la detección del balón, se debe saber que el sensor IRSeeker funciona por medio de señales infrarrojas, devolviendo un número del 1 al 9 dependiendo de donde se encuentra el balón, el cual envía señales infrarrojas. La forma utilizada para tomar el balón consiste en que si le llega como señal al NXT un valor entre 1 y 4, incluyendo estos valores, el robot debe realizar un pequeño giro a la izquierda, si el valor es entre 6 y 9 se debe realizar el giro a la derecha y si el valor obtenido es exactamente 5 debe avanzar, debido a que es el valor que indica que la señal proviene de frente. Una vez se empieza a avanzar se utiliza el sensor de distancia para saber cuándo se tomó el balón.

Una vez terminada la detección y toma de balón se procedió a desarrollar la función con la cual el JugadorA dejara el balón cerca del JugadorB. El encargado de la mayor parte de esta función es el AgenteCV, debido que es el que capta la posición de los triángulos para saber la posición donde se encuentran los NXT. Se optó por utilizar esta forma para obtener el direccionamiento de los robots, porque es menos probable que ocurra una falla en la detección de posición, debido a que si se ocupa el sensor de brújula puede ser que no esté en condiciones óptimas e indique mal las coordenadas.

El AgenteCV le envía las coordenadas de los centros de los dos robots, los cuales son tomadas por medio de un triángulo que se encuentra sobre estos (un triángulo color azul y otro rojo), además del Angulo del triángulo azul con respecto al eje x. Lo primero a verificar cuando se tienen estos datos es si el JugadorA está mirando en la posición opuesta del JugadorB, de no ser así se comienza hacer girar el JugadorA hasta que se consigue que mire en dirección opuesta como muestra la Figura 3.21. Al estar mirando en direcciones opuestas el JugadorA avanza hasta llegar cerca del JugadorB, si se encuentran frente a frente ambos NXT se avanza hasta tener una distancia de 40 cm desde el centro de ambos triángulos, si no se encuentran frente a frentes, se avanza hasta que sus centros estén a una diferencia menor de 5 cm.

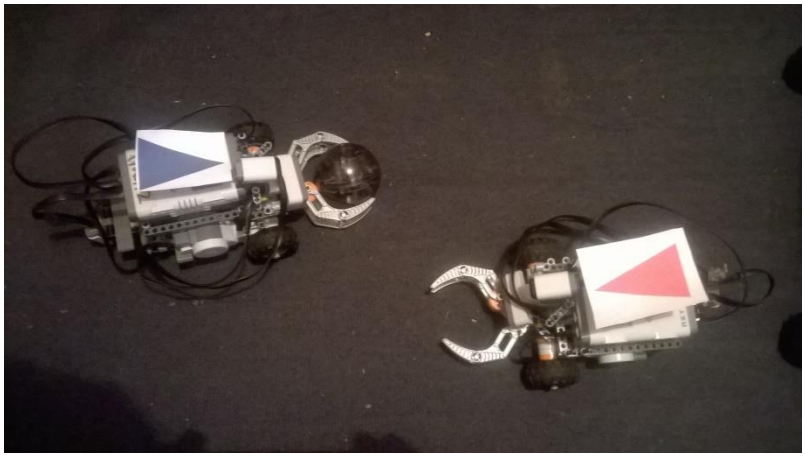

Figura C.3 Dirección de NXTs

Una vez ya en la posición adecuada para dejar el balón el JugadorA suelta el balón y retrocede para así no obstaculizar al JugadorB cuando inicie la toma del balos. Al finalizar el retroceso, el JugadorA envía una señal al AgenteCV indicando que el camino ya está libre para que el JugadorB parta su acción, al recibir esta señal el AgenteCV le informa al JugadorB, el que toma el balón con la misma lógica que la realiza el JugadorA. Al finalizar esta acción el sistema funciono correctamente y deja de funcionar.

C.2 Pruebas

Las pruebas se realizan con el fin de poder detectar los errores y proporcionar información objetiva e independiente sobre la calidad del sistema. Por esta razón en el desarrollo de este avance se realizaron diferentes tipos de prueba a medida que se desarrollaba el sistema.

C.2.1 Entorno de Prueba

El entorno de prueba que fue usado para probar el sistema consiste en:

- Cancha de 87 cm de ancho y 65 cm de largo.
- Cámara fujitel HD de 720p de resolución a una altura de 1,72 metros.
- Balón infrarrojo HiTechnic.
- Dos Lego Mindstorms NXT.

Figura C.4 Entorno de Prueba

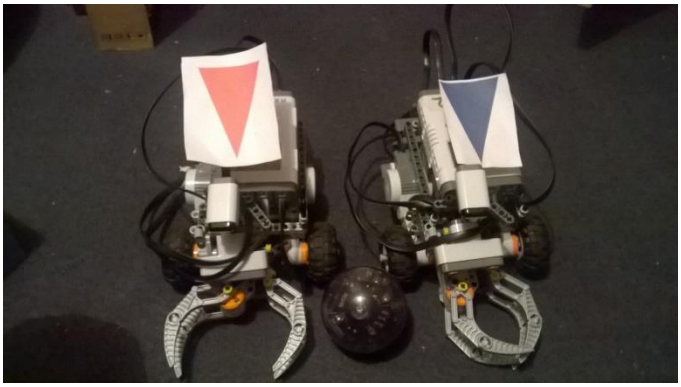

Figura C.5 Lego Mindstorms NXT de prueba y Balón infrarrojo

C.2.2 Prueba de toma de balón

Esta prueba consistió en poder comprobar si el NXT era posible de tomar el balón, por medio de las señales infrarrojas. Para esto se trabajó con un sistema el cual solo realizaba la función que se deseaba probar. Debido a que ambos NXT están construidos de la misma manera solo se comprobó con uno si el sistema funcionaba bien.

Los primeros resultados fueron engañosos, esto debido a que lo primero que se implementó de esta función fue que el NXT girada para mirar el balón de frente, esto resulto bien pero al momento de realizar que se desplazara hacia delante no lo realizaba, porque no indicaba la señal 5 como adelante. Este error ocurrió por una falla en la posición donde se colocó el sensor infrarrojo en el NXT.

Para solucionar la falla nombrada se modificó la posición del sensor probando donde era la mejor posición para colocarlo y que indicada bien las posiciones. Una vez encontrada la posición adecuada el sistema de toma de balón funciono correctamente.

C.2.3 Prueba de Posicionamiento del balón

Esta tenía como finalidad integrar el sistema de detección de triángulos, que se muestra en los Anexos al AgenteCV, para que el JugadorA dejara lo más cerca el balón del jugadorB. Además verificar que los NXT no chocaran entre sí. Para esta prueba se desarrolló un sistema que solo buscaba la posición adecuada donde dejar el balón.

Los primeros resultados no fueron los deseados, debido a que el AgenteCV no enviaba las indicaciones al Agente JugadorA, esto se debió a que no se limpiaba la variable encargada del mensaje produciendo que el sistema se congelada. Una vez solucionado esto ocurrió otro error el cual era que el NXT no se detenía en una posición adecuada, esto ocurrió debido a que la distancia entre los centros de los triángulos que se utilizaba era muy pequeña. Para solucionar el último error se probó con diferentes medidas hasta llegar a una óptima.

C.2.4 Pruebas Sistema completo

El objetivo de esta prueba es juntar todo el sistema en uno solo. Para esto se crearon otras funciones como los son la conexión de los robot, despejar el camino al dejar el balón, entre otras, no fueron consideradas en este punto debido que sus pruebas en la mayoría de los casos fueron exitosas al probarlas una vez o eran cambios muy mínimos para arreglar los errores.

Los resultados de estas pruebas fueron los esperados, esto se debió a que previamente se probaron todas las funciones por separado y solucionando sus errores hasta tener un funcionamiento correcto de estas.

D. Pruebas de latencia

D.1 Tabla prueba de latencia uno

Tabla D.1 Prueba de latencia uno

Mensaje	Envió	Recibo	Latencia
1	1451763907825	1451763907826	1
2	1451763907942	1451763907943	1
3	1451763908063	1451763908064	1
4	1451763908182	1451763908183	1
5	1451763908303	1451763908304	1
6	1451763908424	1451763908425	1
7	1451763908546	1451763908551	5
8	1451763908661	1451763908664	3
9	1451763908785	1451763908789	4
10	1451763908904	1451763908905	1
11	1451763909027	1451763909029	2
12	1451763909144	1451763909148	4
13	1451763909305	1451763909307	2
14	1451763909416	1451763909417	1
15	1451763909515	1451763909518	3
16	1451763909632	1451763909633	1
17	1451763909775	1451763909777	2
18	1451763909864	1451763909866	2
19	1451763909983	1451763909986	3
20	1451763910117	1451763910118	1
21	1451763910223	1451763910224	1
22	1451763910344	1451763910348	4
23	1451763910459	1451763910459	0
24	1451763910583	1451763910584	1
25	1451763910700	1451763910701	1
26	1451763910821	1451763910821	0
27	1451763910942	1451763910943	1
28	1451763911061	1451763911062	1
29	1451763911192	1451763911193	1
30	1451763911300	1451763911301	1

Promedio	1,7
Varianza	1,59655172
Desviación Estándar	1,26354728

D.2 Tabla prueba de latencia dos

Tabla D.2 Prueba de latencia dos

Mensaje	Envió	Recibo	Latencia
1	1451764393693	1451764393694	1
2	1451764393812	1451764393815	3
3	1451764393937	1451764393938	1
4	1451764394052	1451764394053	1
5	1451764394173	1451764394174	1
6	1451764394291	1451764394292	1
7	1451764394412	1451764394413	1
8	1451764394534	1451764394535	1
9	1451764394651	1451764394652	1
10	1451764394771	1451764394772	1
11	1451764394894	1451764394895	1
12	1451764395012	1451764395013	1
13	1451764395134	1451764395135	1
14	1451764395253	1451764395256	3
15	1451764395372	1451764395377	5
16	1451764395493	1451764395494	1
17	1451764395611	1451764395612	1
18	1451764395730	1451764395731	1
19	1451764395850	1451764395851	1
20	1451764395976	1451764395979	3
21	1451764396123	1451764396125	2
22	1451764396265	1451764396268	3
23	1451764396427	1451764396430	3
24	1451764396553	1451764396555	2
25	1451764396759	1451764396760	1
26	1451764396855	1451764396856	1
27	1451764396951	1451764396952	1
28	1451764397060	1451764397061	1
29	1451764397172	1451764397172	0
30	1451764397294	1451764397294	0
31	1451764397411	1451764397412	1
32	1451764397531	1451764397532	1
33	1451764397652	1451764397653	1
34	1451764397772	1451764397773	1
35	1451764397895	1451764397896	1

36	1451764398013	1451764398014	1
37	1451764398131	1451764398132	1
38	1451764398252	1451764398253	1
39	1451764398372	1451764398373	1
40	1451764398566	1451764398567	1
41	1451764398660	1451764398660	0
42	1451764398754	1451764398755	1
43	1451764398853	1451764398854	1
44	1451764398971	1451764398972	1
45	1451764399092	1451764399093	1
46	1451764399215	1451764399216	1
47	1451764399340	1451764399341	1
48	1451764399451	1451764399451	0
49	1451764399571	1451764399572	1
50	1451764399692	1451764399692	0
51	1451764399815	1451764399816	1
52	1451764399931	1451764399932	1
53	1451764400052	1451764400053	1
54	1451764400171	1451764400172	1
55	1451764400290	1451764400291	1
56	1451764400414	1451764400415	1
57	1451764400531	1451764400532	1
58	1451764400651	1451764400652	1
59	1451764400770	1451764400770	0
60	1451764400896	1451764400897	1
61	1451764401011	1451764401012	1
62	1451764401133	1451764401133	0
63	1451764401250	1451764401251	1
64	1451764401371	1451764401371	0
65	1451764401493	1451764401494	1
66	1451764401610	1451764401611	1
67	1451764401731	1451764401732	1
68	1451764401852	1451764401852	0
69	1451764401971	1451764401972	1
70	1451764402090	1451764402091	1
71	1451764402216	1451764402217	1
72	1451764402335	1451764402336	1
73	1451764402452	1451764402453	1
74	1451764402573	1451764402573	0
75	1451764402691	1451764402692	1

76	1451764402814	1451764402815	1
77	1451764402932	1451764402933	1
78	1451764403097	1451764403098	1
79	1451764403187	1451764403187	0
80	1451764403292	1451764403293	1
81	1451764403411	1451764403411	0
82	1451764403531	1451764403532	1
83	1451764403651	1451764403651	0
84	1451764404012	1451764404013	1
85	1451764404131	1451764404132	1
86	1451764404250	1451764404251	1
87	1451764404372	1451764404373	1
88	1451764404492	1451764404493	1
89	1451764404613	1451764404614	1
90	1451764404732	1451764404733	1
91	1451764404851	1451764404851	0
92	1451764404974	1451764404975	1
93	1451764405091	1451764405092	1
94	1451764405211	1451764405212	1
95	1451764405331	1451764405332	1
96	1451764405453	1451764405454	1
97	1451764405576	1451764405576	0
98	1451764405811	1451764405812	1
99	1451764405932	1451764405933	1
100	1451764406053	1451764406054	1

Promedio	1,01
Varianza	0,53525253
Desviación Estándar	0,73160954

D.3 Tabla prueba de latencia tres

Tabla D.3 Prueba de latencia tres

Mensaje	Envió	Recibo	Latencia
1	1451764573217	1451764573218	1
2	1451764573337	1451764573337	0
3	1451764573456	1451764573459	3
4	1451764573579	1451764573580	1
5	1451764573701	1451764573703	2
6	1451764573819	1451764573820	1

7	1451764573938	1451764573942	4
8	1451764574057	1451764574061	4
9	1451764574421	1451764574425	4
10	1451764574548	1451764574550	2
11	1451764574662	1451764574664	2
12	1451764574782	1451764574784	2
13	1451764574909	1451764574911	2
14	1451764575019	1451764575020	1
15	1451764575139	1451764575140	1
16	1451764575274	1451764575278	4
17	1451764575379	1451764575380	1
18	1451764575618	1451764575619	1
19	1451764575737	1451764575738	1
20	1451764575980	1451764575981	1
21	1451764576097	1451764576098	1
22	1451764576218	1451764576219	1
23	1451764576457	1451764576463	6
24	1451764576702	1451764576703	1
25	1451764576819	1451764576820	1
26	1451764576939	1451764576940	1
27	1451764577062	1451764577063	1
28	1451764577181	1451764577182	1
29	1451764577298	1451764577298	0
30	1451764577423	1451764577424	1
31	1451764577659	1451764577660	1
32	1451764577780	1451764577781	1
33	1451764577898	1451764577899	1
34	1451764578021	1451764578021	0
35	1451764578142	1451764578143	1
36	1451764578258	1451764578259	1
37	1451764578377	1451764578378	1
38	1451764578498	1451764578499	1
39	1451764578618	1451764578619	1
40	1451764578738	1451764578739	1
41	1451764578859	1451764578860	1
42	1451764578976	1451764578977	1
43	1451764579099	1451764579100	1
44	1451764579217	1451764579218	1
45	1451764579339	1451764579339	0
46	1451764579457	1451764579458	1

47	1451764579579	1451764579580	1
48	1451764579697	1451764579698	1
49	1451764579816	1451764579817	1
50	1451764579936	1451764579937	1
51	1451764580057	1451764580058	1
52	1451764580176	1451764580177	1
53	1451764580297	1451764580298	1
54	1451764580415	1451764580416	1
55	1451764580656	1451764580657	1
56	1451764580776	1451764580777	1
57	1451764581135	1451764581136	1
58	1451764581738	1451764581739	1
59	1451764581977	1451764581978	1
60	1451764583055	1451764583056	1
61	1451764583776	1451764583776	0
62	1451764584014	1451764584015	1
63	1451764584135	1451764584135	0
64	1451764584256	1451764584257	1
65	1451764584735	1451764584736	1
66	1451764585214	1451764585215	1
67	1451764585574	1451764585575	1
68	1451764585815	1451764585816	1
69	1451764588458	1451764588459	1
70	1451764588580	1451764588581	1
71	1451764588698	1451764588699	1
72	1451764588818	1451764588818	0
73	1451764588946	1451764588947	1
74	1451764589055	1451764589056	1
75	1451764589176	1451764589176	0
76	1451764589296	1451764589297	1
77	1451764589417	1451764589418	1
78	1451764589548	1451764589548	0
79	1451764589749	1451764589750	1
80	1451764589844	1451764589845	1
81	1451764589941	1451764589942	1
82	1451764590040	1451764590041	1
83	1451764590137	1451764590138	1
84	1451764590257	1451764590258	1
85	1451764590380	1451764590381	1
86	1451764590497	1451764590498	1

87	1451764590624	1451764590625	1
88	1451764590744	1451764590745	1
89	1451764590864	1451764590865	1
90	1451764590986	1451764590987	1
91	1451764591106	1451764591107	1
92	1451764591224	1451764591225	1
93	1451764591341	1451764591342	1
94	1451764591461	1451764591463	2
95	1451764591583	1451764591584	1
96	1451764591704	1451764591704	0
97	1451764591824	1451764591825	1
98	1451764591940	1451764591941	1
99	1451764592060	1451764592061	1
100	1451764592180	1451764592181	1

Promedio	1,15
Varianza	0,79545455
Desviación Estándar	0,89188259