ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA ELECTRÓNICA Y AUTOMATIZACIÓN

"IMPLEMENTACIÓN DE UN SISTEMA APOYO A LA EDUCACIÓN PARVULARIA CON UN ROBOT HUMANOIDE UTILIZANDO TÉCNICAS DE VISIÓN ARTIFICIAL"

Trabajo de titulación:

Tipo: Propuesta Tecnológica

Presentado para optar al grado académico de:

INGENIERA EN ELECTRÓNICA Y AUTOMATIZACIÓN

AUTORA:

RUTH VIVIANA CASTAÑEDA COYAGO

Riobamba – Ecuador

2021

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA ELECTRÓNICA Y AUTOMATIZACIÓN

"IMPLEMENTACIÓN DE UN SISTEMA APOYO A LA EDUCACIÓN PARVULARIA CON UN ROBOT HUMANOIDE UTILIZANDO TÉCNICAS DE VISIÓN ARTIFICIAL"

Trabajo de titulación:

Tipo: Propuesta Tecnológica

Presentado para optar al grado académico de:

INGENIERA EN ELECTRÓNICA Y AUTOMATIZACIÓN

AUTORA: RUTH VIVIANA CASTAÑEDA COYAGO

DIRECTOR: Ing. JOSÉ LUIS MORALES GORDÓN Mgs.

Riobamba – Ecuador

2021

Yo, **Ruth Viviana Castañeda Coyago**, declaro que el presente trabajo de titulación es de mi autoría y los resultados de este son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citadas y referenciadas.

Como autora asumo la responsabilidad legal y académica de los contenidos de este trabajo de titulación, el patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 20 de diciembre de 2021

Ruth Viviana Castañeda Coyago

175036993-4

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA ELECTRÓNICA Y AUTOMATIZACIÓN

El Tribunal del Trabajo de Integración Curricular certifica que: El trabajo de Integración Curricular; Tipo**:** Propuesta Tecnológica, **IMPLEMENTACIÓN DE UN SISTEMA APOYO A LA EDUCACIÓN PARVULARIA CON UN ROBOT HUMANOIDE UTILIZANDO TÉCNICAS DE VISIÓN ARTIFICIAL**, realizado por la señorita **RUTH VIVIANA CASTAÑEDA COYAGO** ha sido minuciosamente revisado por los Miembros del Tribunal del Trabajo de Integración Curricular, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal autoriza su presentación.

	FIRMA	**FECHA**
Ing. Fausto Ramiro Cabrera Aguayo Mgs. **DIRECTOR DEL TRIBUNAL**	FAUSTO RAMIRO CABRERA AGUAYO Digitally signed by FAUSTO RAMIRO CABRERA AGUAYO Date: 2022.01.03 19:59:14 -05'00'	2021-12-20
Ing. José Luis Morales Gordón Mgs. **DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR**	Jose Morales Firmado digitalmente por Jose Morales Fecha: 2022.01.07 16:19:55 -05'00'	2021-12-20
Ing. Pablo Eduardo Lozada Yánez Mgs. **MIEMBRO DEL TRIBUNAL**	Firmado digitalmente por PABLO EDUARDO LOZADA YANEZ	2021-12-20

DEDICATORIA

El presente trabajo de titulación primero lo dedicó a Dios, por cuidarme, protegerme y darme la fuerza y sabiduría para poder terminar esta etapa de mi vida, luego a mis padres Francisco y Mariana, por sacrificarse cada día para darme lo mejor, son el apoyo más grande que tuve en este camino y sin ustedes no hubiera sido posible llegar tan lejos, son el regalo perfecto de Dios, los quiero mucho, luego a mis hermanos Robin, Laura, Juan y Mariana, por recordarme que puedo contar con ellos siempre, a mis sobrinos por permitir ser ejemplo en sus vidas, y como último a mis amigos que hicieron alegre y amena esta etapa de mi vida, gracias por estar conmigo en los buenos y malos momentos. Para todos ustedes quiero dedicar este trabajo con mucho cariño y amor.

Ruth Viviana

AGRADECIMIENTO

El más sincero agradecimiento a la Escuela Superior Politécnica de Chimborazo y a la Facultad de Informática y Electrónica, por darme la oportunidad de obtener una profesión y hacer una investigación de ayuda a la sociedad, el agradecimiento a cada uno de mis profesores por compartir sus conocimientos a lo largo de la carrera.

A mi incondicional familia por su apoyo y comprensión.

Ruth Viviana

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE ECUACIONES

ÍNDICE DE ANEXOS

ÍNDICE DE ABREVIATURAS

RNA: Red Neuronal Artificial

OPENCV: Visión Artificial Abierta por Computadora

IA: Inteligencia Artificial

RNC: Red Neuronal Convolucional

RBF: Red de Base Radial

RNN: Red Neuronal Recurrente

MVP: Máquina Virtual Python

SDK: Kid de Desarrollo de Software

SDK-Naoqi: Software de Desarrollo del Robot Nao

MC: Matriz de Confusión

HAAR: Funciones rectangulares en dos dimensiones

GUI: Interfaz de usuario gráfica

CM-530: Controlador del Robot Bioloid GP

RC-100A: Control remoto del Bioloid GP

Cm: centímetro

USB: Bus Universal en Serie

HDMI: Interfaz multimedia de Alta Definición

HMI: Interfaz Hombre-Máquina

GDL: Grados de Libertad

Km: Kilómetros

V: Voltaje

Ah: Amperio-hora

A: Amperio

Wh: Vatio-hora

Min: Minutos

Px: pixeles

MP: Mega pixeles

µm: Micrómetro

dB: Decibelio

lux-seg: Luxmetro-segundo

fps: Fotograma por segundo

DFOV: Campo Visual Diagonal

VFOV: Ventilación Oscilatoria de Alta Frecuencia

KHz: Kilohercio

RAM: Memoria de Acceso Aleatorio

HxV: Horizontal por Vertical

GB: Gigabyte

XML: Lenguaje Marcado Extensible

RGB: Rojo, verde, azul

HSV: Matiz, saturación, valor

Flashcard: Tarjeta de presentación

mV/Pa: Milivoltio por Pascal

RESUMEN

En la presente investigación se implementó un sistema apoyo a la educación parvularia con un robot humanoide utilizando técnicas de visión artificial denominado *Happy*, se seleccionó al robot Nao el cual cumplió con los requerimientos de operatividad y funcionalidad a nivel de hardware y software. El sistema consistió en enseñarles a niños de 4 años colores y figuras geométricas a través del manejo de *flashcards*, para esto se usó técnicas de visión artificial como: preprocesamiento, segmentación y extracción de características, donde los algoritmos de aprendizaje supervisado se entrenaron con redes neuronales convolucionales y clasificadores tipo HARR en un núcleo de red en cascada, con esto el sistema tuvo la capacidad de identificar imágenes por color y figura geométrica. En base a las pruebas del detector de color, detector de figura y detector de figura con color en ambientes controlados y no controlados, se determinó que los modelos entrenados tienen una precisión de 95% en cuadrados, 92% en círculos y 91% en triángulos y una aceptación del robot Nao de 95,12% por los niños. Se concluye que el sistema *Happy* es de ayuda en la formación cognitiva de niños al ser una herramienta amigable que capta su atención, entrena su memoria visual y despierta su curiosidad. Se recomienda que se sigan haciendo investigaciones que involucren a la robótica en la educación, para en un futuro innovar el pensum pedagógico de instituciones educativas.

Palabras clave: <TECNOLOGÍA Y CIENCIAS DE LA INGENIERÍA> <ROBOT HUMANOIDE> <INTELIGENCIA ARTIFICIAL> <REDES NEURONALES> <EDUCACIÓN PARVULARIA>.

SUMMARY

In this research work, a support system for preschool education was implemented with a humanoid robot using artificial vision techniques called Happy, the Nao robot was selected which met the operational and functional requirements at hardware and software level. The system consisted of teaching colors and geometric shapes to 4-year-olds through the use of flashcards, for this artificial vision techniques were used such as: preprocessing, segmentation and feature extraction, where the algorithms of supervised learning were trained with convolutional neural networks and classifiers of HARR type in a cascade network core; with this, the system had the ability to identify images by color and geometric shape. Based on tests of color detector, shape detector and colored-shape detector in controlled and uncontrolled environments, it was determined that trained models are 95% accurate in squares, 92% in circles, and 91% in triangles and an acceptance of the Nao robot of 95.12% by the children. It is concluded that the system Happy is helpful in the cognitive training of children as it is a friendly tool that captures their attention, trains their visual memory and arouses their curiosity. It is recommended to keep conducting research that involves robotics in education, to innovate the pedagogical curriculum of educational institutions in the future.

Keywords: <TECHNOLOGY AND ENGINEERING SCIENCES> <HUMANOID ROBOT> <ARTIFICIAL INTELLIGENCE> <NEURONAL NETWORKS> <PRESCHOOL EDUCATION>.

INTRODUCCIÓN

La robótica educativa tiene sus orígenes desde 1960 con un lenguaje de programación denominado *Logo,* el creador *Seymour Papert* lo implementó con un robot Tortuga para seguir una lista de instrucciones dirigida por comandos al formar figuras geométricas, por medio de esta base se crearon diferentes tipos de robots para involucrarse en la formación cognitiva de niños y jóvenes. En países de Europa y Asia en los pensum de estudio se encuentra la robótica inmersa como un programa educativo al trabajar con robots en aulas de clases para formar habilidades y destrezas en áreas de ciencia y tecnología. (Moreno et al., 2012: pp.76-79)

La robótica en la educación parvularia desarrolla un ambiente de trabajo interdisciplinario, usa tecnologías que permiten al estudiante adquirir destrezas al construir conceptos y resolver problemas. El avance de la robótica en la educación ha innovado desde kits como: *WeDo* hasta robots humanoides con plataformas programables, los cuales son capaces de involucrarse en aulas de primaria, secundaria, universidad e institutos para proporcionan al docente herramientas de apoyo didácticas adaptadas a nuevas innovaciones. (Hervás et al., 2018: pp.30-34)

En la propuesta tecnológica se implementa un sistema apoyo a la educación parvularia con un robot humanoide utilizando técnicas de visión artificial, en el cual se desarrolla un ambiente de aprendizaje con el uso de la robótica en las aulas de clases, la pedagogía cotidiana no se cambia solo se incluye un plus innovador en el proceso de formación académica.

ANTECEDENTES

El término párvulo procede del latín *"parvulus"* el cual hace referencia a un niño pequeño, en el ámbito educativo son estudiantes entre 3 a 5 años que comienzan a desarrollar habilidades socioemocionales, cognitivas y psicomotrices. A la educación parvularia se la caracteriza por ser una parte esencial en la formación del ser humano, donde todo lo aprendido y adquirido se convierten en cimientos de su desarrollo intelectual. (Castello et al., 1995: p.12)

El ministerio del Ecuador en la constitución del año 2008 en su artículo 344, menciona a la educación parvularia como el nivel primario del sistema educativo, donde se desarrollan destrezas para pensar, razonar y decidir, no obstante, existe la oportunidad para promover la confianza y autoestima en el niño. La ley orgánica de educación del Ecuador publicada en el año 2011 del mes de marzo hace responsables de la formación de los pequeños al estado, la familia, la comunidad y centros de aprendizaje, los cuales son encargados de proveer recursos acordes a los cambios tecnológicos. (Ministerio de Educación del Ecuador, 2012, pp.32-35)

La educación está cambiando y los niños en la actualidad tienen presente a la tecnología, las pedagogías tradicionales comienzan a ser obsoletas y ya no se pretende enseñar a niños a memorizar conceptos, sino a formar su propio conocimiento a través del pensamiento crítico, la robótica dentro de la educación le está dando un giro, donde se pasa de las clases habituales a otras que incluyen robots con nuevas técnicas de enseñanza-aprendizaje. (Martínez, 2017, p.3)

A nivel mundial los robots ya son parte de las nuevas pedagogías académicas, en la *Escuela Paul Bert de Poitiers de Francia* el robot Nao es encargado de enseñar gramática a los niños con el uso de recursos audiovisuales para tener su atención, el profesor que supervisa esta metodología se llama *Serge Evrinoff,* el cual afirma que usar robots en las aulas ayuda a tener un mayor dinamismo con los niños y los hace más sensibles a los avances de la informática, otro es el *Instituto Shoshi de Waseda en Japón*, donde el robot Pepper es asistente del profesor de lengua y esta capacitado para contestar las dudas sobre lo que se impartió en clase a través de sus recursos de audio y video. (Goméz, 2018, p.2)

Al tener un robot en la educación se ha comprobado que puede generar cambios positivos en el desarrollo académico de un niño o un joven, a nivel mundial en muchos Institutos ya son parte de la vida educativa de un estudiante, sin embargo, en Ecuador aún no son una realidad estable, las investigaciones sobre algoritmos de inteligencia artificial en los robots puedan ayudar a contribuir que en un futuro sean una realidad evidente y se conviertan en un sistema de apoyo a la educación.

PLANTEAMIENTO DEL PROBLEMA

¿Cómo implementar un sistema apoyo a la de educación parvularia con un robot humanoide utilizando técnicas de visión artificial?

Sistematización del problema

- ¿Cuál es la operatividad y funcionalidad de usar un robot humanoide en la educación parvularia con técnicas de visión artificial?
- ¿Cuáles son los requerimientos que debe tener un robot humanoide para realizar un sistema apoyo a la de educación parvularia utilizando técnicas de visión artificial?
- ¿Cuáles son las técnicas de visión artificial que permiten crear un algoritmo de enseñanza de párvulos en robots humanoides?
- ¿Como se puede implementar un algoritmo de enseñanza con visión artificial en un robot humanoide, para que sea un sistema apoyo a la educación parvularia y tener una interacción robot-niño?

- ¿Cómo se puede evaluar un sistema apoyo en la de educación parvularia con un robot humanoide con técnicas de visión artificial?

JUSTIFICACIÓN DEL PROYECTO

El Currículo de Educación inicial del año 2014 del Ecuador resalta la importancia que tiene la formación de los niños de 3 a 5 años, se los reconoce como seres bio-psicosociales y los sitúa como uno de los principales ejes en la educación, la ley orgánica de Educación Intercultural (LOEI), en el artículo 22 literal c), designa a la Autoridad Nacional como el encargado de hacer cumplir el derecho a la educación de los niños, donde deben buscar que tengan una educación de calidad al proporcionar tecnología y herramientas necesarias para mejorar su desempeño académico dentro del aula. (Ministerio de Educación del Ecuador, 2012, pp.15-16)

La robótica en la educación ha surgido en los últimos años como una propuesta innovadora, a nivel mundial diversos planteles manejan este nuevo sistema y tienen buenos resultados al salir de las rutinas educativas. En Ecuador no es tan real utilizar robots humanoides en las aulas de clases, pero se necesita dar a conocer a la población los diferentes avances que existen en otros países y los beneficios que genera el innovar metodologías de enseñanza con el uso de la tecnología. (Monsalves, 2010, pp.90-92)

OBJETIVOS

Objetivo general

Implementar un sistema apoyo a la de educación parvularia con un robot humanoide utilizando técnicas de visión artificial.

Objetivos específicos

- Investigar la operatividad y funcionalidad de usar un robot humanoide en la educación parvularia con técnicas de visión artificial.
- Establecer los requerimientos que debe tener un robot humanoide para poder aplicar un sistema de visión artificial y aprendizaje automático para la interacción robot-humano.
- Seleccionar el software y el hardware que se va a utilizar, y las técnicas de visión artificial que se van a emplear en el robot humanoide.
- Implementar un algoritmo de aprendizaje con visión artificial, para que sea un sistema apoyo a la educación parvularia.

- Evaluar el sistema de apoyo a la educación parvularia con el robot humanoide utilizando visión artificial, con el fin de verificar la funcionalidad.

CAPÍTULO I

1 MARCO TEÓRICO

En el presente capítulo se define la robótica y se analiza los avances tecnológicos que se conocen sobre los diferentes tipos de robots humanoides comerciales, y como se desarrollan en la educación. Después se analiza la relación de la inteligencia artificial con la robótica y las técnicas de visión artificial.

1.1 Robótica

La robótica es una ciencia que estudia la mecánica de los robots y se la conoce por unir la electrónica, la informática y la ingeniería en un solo módulo al automatizar un sistema virtual para controlar procesos en diferentes fases, esta área evoluciona a partir de los avances técnicos que involucra a la inteligencia artificial como una base en la creación de módulos autónomos con escenarios de interacción humano-robótica. (López et al., 2013: pp.44-45)

Los tipos de robots humanoides desarrollados en los últimos años han cambiado la perspectiva de vida, tanto en el ámbito laboral como educativo, en donde se generó módulos autómatas capaces de realizar actividades que requieran un exceso de esfuerzo físico, no obstante, con la intervención de la inteligencia artificial esta tecnología se ubicó en un siguiente nivel con el diseño de algoritmos para observar, procesar y decidir. (Cabás, 2009, pp.8-11)

1.1.1 Robots humanoides

Los robots humanoides se distinguen por tener una apariencia humana, su sistema se enfoca en el uso de la inteligencia artificial, para lograr adaptarse a diferentes entornos de desarrollo, como: espacios de estudio y trabajo, la arquitectura de estos robots imita el comportamiento del humano y aprende de él, al generar trayectorias con acciones secuenciales basados en la percepción de datos preseleccionados, con esto se define un patrón de acción en la respuesta del humanoide. (Cabás, 2009, pp.8-11)

Alrededor del mundo diferentes industrias se han dedicado a la investigación y creación de robots humanoides, los diseños en la actualidad más conocidos son Bioloid GP, DARwin-Op, Pepper, Nao, Atlas y Sophia, aparte de estos existen robots que se han desarrollado en Universidades, pero a diferencia de estos no son comerciales y no tienen una plataforma formal de programación accesible. A continuación, se detalla las especificaciones técnicas de cada uno de los robots humanoides mencionados. (Rodríguez, 2018, pp.13-15)

1.1.1.1 Robot Bioloid GP

Es un robot humanoide educativo diseñado por la empresa *ROBOTIS PREMIUM*, tiene un software programable a través de un controlador CM-530 con una estructura metálica de 16 grados de libertad compuesto por 18 servomotores *DYNAMIXEL* y una altura de 35cm, el diseño se visualiza en la figura 1-1, puede ser dirigido por un control remoto RC-100A, o de manera autónoma, la interfaz de programación está en un lenguaje estructurado con secuencia de patrones para manipular la locomoción bípeda y configurar un sistema propio en el robot. (Cabrera et al., 2016: pp.43-47)

Figura 1-1: Robot Humanoide Bioloid GP
Fuente: Cabrera et al., 2016

1.1.1.2 Robot DARwin-Op

Es un robot humanoide educativo diseñado por la empresa *ROMELA LAB*, en conjunto con la Universidad de Pensylvania, tiene un procesador Intel *Atom* Z530 de 1.6GHz, un sistema de audio y video, una estructura modular que puede ser modificada, una conexión vía Ethernet, USB o HDMI y una altura de 45.5cm en 20 grados de libertad, el diseño se visualiza en la figura 2-1. El lenguaje de programación es a través de Java, C++, Python, Labview o Matlab, se lo utiliza para trabajar en aplicaciones que impliquen dinámica del robot o procedimientos de control automático y con esto crear módulos educativos de enseñanza-aprendizaje. (Barrera, 2016, pp.22-24)

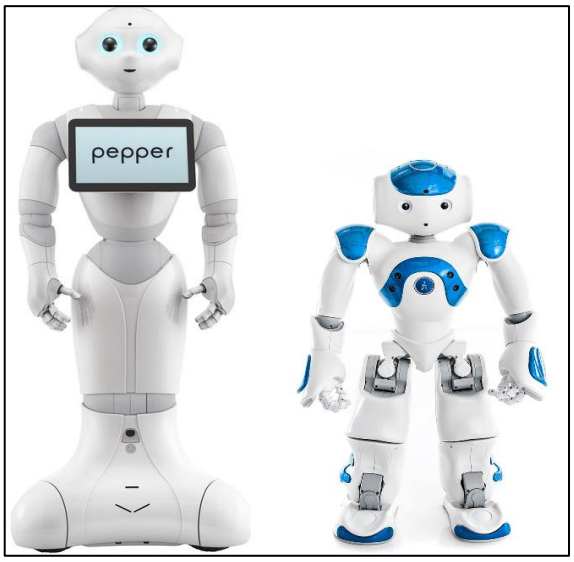
Figura 2-1: Robot Humanoide DARwin-Op

Fuente: https://www.robotic-lab.com/blog/uploads/2011/07/Darwin_OP

1.1.1.3 Robot Pepper y Nao

Son robots humanoides desarrollados por *SoftBank Robotics*, Nao tienen una conexión vía Ethernet y Wifi, con un diseño amigable de policarbonato que se visualiza en el literal b) de la figura 3-1, tiene una altura de 58cm, con 9 sensores táctiles, 8 de presión, 2 ultrasónicos y dos cámaras en la parte frontal de la cabeza, en cambio Pepper tiene una altura de 120cm con una cámara 3D, cuatro micrófonos, una pantalla táctil y una velocidad de desplazamiento de 3km/h, el diseño se visualiza en literal a) de la figura 3-2. (ALIVEROBOTS,2018)

La interfaz de programación de estos humanoides es a través de *Chereographe*, C++ y Python, se puede trabajar con algoritmos de inteligencia artificial y sistemas con redes neuronales, para tener módulos operativos autónomos, el robot Pepper a diferencia de Nao con la pantalla táctil comparte datos de gestión de una empresa y maneja una tienda online corporativa al mostrar proyectos con tipificaciones operantes, en cambio Nao es incluido en sistemas educativos para interactuar con estudiantes y crear módulos de enseñanza . (ALIVEROBOTS,2018)



a) Pepper b) Nao

Figura 3-1: Robots Humanoides a) Pepper b) Nao

Fuente: ALIVEROBOTS,2018

1.1.1.4 Robot Atlas

Es un robot Humanoide desarrollado por *Boston Dynamics*, mide 1,8m con un diseño de aluminio aeronáutico que se visualiza en la figura 4-1 distribuidos en 25 grados de libertad, son capaces de identificar objetos y realizar movimientos complejos, son destinados a estar en ambientes de peligro con condiciones inestables, tienen un gran equilibrio para realizar saltos de hasta 40cm y desplazamientos largos, se usa en equipos de rescate que requieren habilidad física, a pesar de ser robots comerciales no tienen un software libre de programación, por lo cual su acceso es limitado. (BOSTONDYNAMICS)

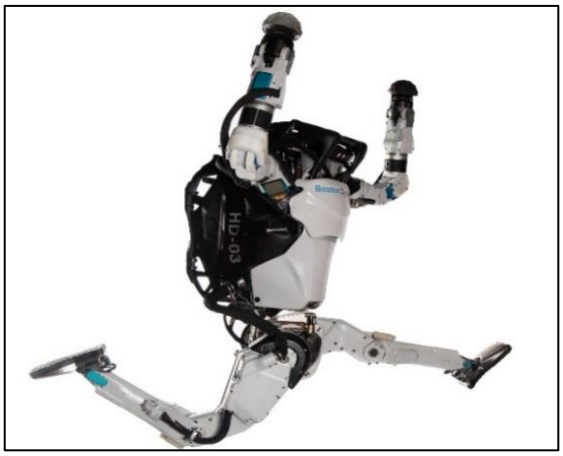


Figura 4-1: Robot Atlas

Fuente: https://www.bostondynamics.com/atlas

1.1.1.5 Robot Sophia

Es un robot humanoide desarrollado por *David Hanson*, creador de la empresa *Hanson Robotics*, son robots capaces de tener conversaciones interactivas, reconocer rostros, expresiones faciales, y tener movimientos articulados, su altura es de 1.65m con 74 grados de libertad en un diseño de resina que se visualiza en la figura 5-1, su conexión es través de USB, HDMI, y Wifi, tiene integrado un patrón de emociones y un sistema para trabajar con ROS, su lenguaje de programación es diseñado para investigaciones que involucren la inteligencia artificial y la robótica. (Buriticá, 2020)

Figura 5-1: Robot Sophia

Fuente: https://www.robotshop.com/us/es/robot-humanoide-sophia-2020-version-rd-hanson-robotics.html

1.1.2 Tabla comparativa de Robots Humanoides

Después de detallar las especificaciones técnicas de los robots humanoides mencionados, se realiza una tabla comparativa para determinar las ventajas y diferencias de utilizar un determinado robot, además se analiza los costos de cada uno en la industria. Con estos datos se identifica el tipo de robot que cumple con los requisitos del sistema tanto en hardware como software. A continuación, en la tabla 1-1, se detalla las características principales de los robots humanoides estudiados, el precio y su arquitectura a través de una tabla recopiladora de datos.

Tabla 1-1: Tabla comparativa de Robots Humanoides

Característica	TIPOS DE ROBOTS HUMANOIDES					
	Bioloid GP	**DarwinOP**	**Nao**	**Pepper**	**Sophia**	**Atlas**
Empresa	ROBOTIS PREMIUM	ROMELA LAB	SOFTBANK-ROBOTICS	SOFTBANK-ROBOTICS	HANSON ROBOTICS	BOSTON DYNAMICS
Material	Aluminio	Aluminio	Policarbonato	Policarbonato	Resina	Aluminio aeronáutico
Altura	35cm	45,5cm	58cm	120cm	1,65m	1,80m
Peso	1,6kg	3kg	4,3kg	28kg	21kg	150kg
Grados de libertad	16 GDL	20 GDL	25 GDL	20 GDL	74 GDL	28 GDL
Conexión	Puerto USB HDMI	Puerto USB HDMI	Wifi Ethernet	Wifi Ethernet	Wifi Ethernet	Wifi Ethernet
Enfoque	Robots educativos para trabajos de locomoción bípeda	Robots educativos para diseñar sistemas con audio y video	Robots educativos y de investigación, tienen un procesador que permite desarrollar módulos de IA	Robots empresariales encargados en gestión de proyectos	Robot con plataformas de investigación dirigidas a la IA	Robot rescatista, que se adapta a condiciones extremas de trabajo
Precio	$ 2 399,90	$ 10 595,00	$ 9 500,00	$ 30 000,00	$ 80 000,00	Aproximado$ 74 500,00

Realizado por: Castañeda, Ruth V., 2021

En la tabla 1-1, se tiene 6 tipos de robots humanoides, de los cuales 4 tienen una conexión vía Wifi, 3 son destinados para sistemas educativos con plataformas programables y otros tres en investigaciones, el costo de estos varía según las especificaciones técnicas y el nivel del procesador, con este análisis se considera que DARwin-Op, Nao y Sophia son aptos para desarrollar el sistema, debido a que son robots destinados para trabajar en sistemas educativos con investigaciones en IA, sin embargo para la elección se considera costos, accesibilidad y operatividad.

1.2 Robótica Educativa

La robótica educativa es conocida como ingeniería pedagógica, involucra al estudiante en su aprendizaje por medio de actividades de exploración, descubrimiento y curiosidad, trabajar con robots en las aulas de clases no tiene como fin reemplazar al docente, sino integrar un asistente e

interfaz entre ellos para ayudar en los procesos de enseñanza-aprendizaje y obtener nuevas técnicas de desarrollo cognitivo. (Barrera, 2015 pp.218-220)

Los métodos de aprendizaje a medida que los descubrimientos avanzan también se reinventan, es necesario hacerlo desde temprana edad para formar el carácter y afianzar los conocimientos, la robótica educativa al involucrarse en esta fase permite al estudiante tener una acción participativa en el aula y crear nuevos criterios en base a su razonamiento. (Barrera, 2015 pp.218-220)

1.2.1 Método de Flashcards en la robótica

Uno de los métodos más conocidos en la pedagogía con niños es el uso de *flashcards* o tarjetas de presentación, son herramientas didácticas de aprendizaje a través de atracción visual, captan la atención del estudiante con el uso de colores, figuras o dibujos en las tarjetas, la enseñanza de los niños cuando no es clara en un lenguaje verbal, una de las opciones es esta metodología, que activa la memoria visual para transmitir conceptos y crear criterios con su razonamiento. En la robótica educativa se hace uso de esta pedagogía con el aprendizaje de idiomas, de números y figuras geométricas, donde los robots juntos con los niños identifican las imágenes para crear juegos de memoria. (Peñafiel et al., 2015: pp.38-40)

1.2.2 Aprendizaje con Robots Humanoides

El uso de Robots Humanoides en las aulas de clases da una versatilidad en la educación, en donde se desarrolla una empatía en los estudiantes por la tecnología, con los ejercicios físicos e intelectuales los incentivan a incluirse en las actividades y superar la timidez por la confianza al mejorar sus destrezas sociales. (SOFTBANK, 2019)

1.2.3 Proyectos Desarrollados

En la Universidad Nacional de Educación en España se han realizado estudios con niños que poseen trastorno de autismo para ayudarlos a integrarse en los programas de enseñanza-aprendizaje con el robot Nao por medio de una aplicación denominada *Autism Solution for Kids (ASK),* que le da respuestas concretas al niño evitando que tenga conflictos emocionales, y a su vez como un mediador social entre los compañeros de clases, una de las ventajas de usar este robot es presentar el aprendizaje como un juego interactivo, ver figura 6-1. (SOFTBANK, 2019)

Figura 6-1: Aprendizaje con el Robot Nao

Fuente: SOFTBANK, 2019

En el *Tecnical College* de la Universidad de Londres, se maneja al Robot Pepper como un sistema de apoyo a la enseñanza con niños de 4 a 8 años, las herramientas del robot crean clases participativas con pedagogías de desarrollo en competencias cognitivas al involucrar conceptos de ingeniería y tecnología en las actividades, con los resultados que tuvieron consideraron a este método amigable y adecuado en niños que les cuesta comunicarse ver figura 7-1. (Vidal, 2016)

Figura 7-1: Aprendizaje con el Robot Pepper

Fuente: SOFTBANK, 2019

1.3 Inteligencia Artificial

La inteligencia artificial es la capacidad que tienen los sistemas para aprender de los datos al usar algoritmos de *Machine Learning*, es un sistema capaz de tomar decisiones como las de un ser humano, detectar objetos, reconocer voces y comprender el procesamiento natural del lenguaje, con los datos de entrada importados establece patrones de comportamiento en un método de aprendizaje profundo o automático, no obstante la IA se utiliza en la resolución de problemas complejos donde se requiere tener segmentos con una percepción similar a las acciones de un humano que involucren; ver, oír y comprender. (Rouhiainen, 2018, pp. 17-24)

La IA dentro de la robótica diseña algoritmos con bases sólidas de automatización, los sistemas artificiales generados implementan métodos de aprendizaje concurrentes para tener una respuesta con perspectiva humana, con esto los diseños de robots dejaron de ser simples y se convirtieron en módulos complejos de interfaz HMI, esto convierte a la IA en un punto principal de perfeccionamiento en la robótica. (Reyes, 2015, pp.46-50)

1.3.1 Visión Artificial

La Visión Artificial es un sistema autónomo con algoritmos capaces de representar la percepción de un humano en procesos de toma de decisiones, la imagen que ve la captura, la segmenta y toma patrones significativos para compararlos con la información de los datos estructurados, el resultado de la interpretación que tiene depende del algoritmo de visión entrenado y la magnitud de datos seleccionados al extraer la información inicial. (Rouhiainen, 2018, pp. 17-24)

La Visión Artificial en la robótica diseña sistemas de decisión a partir de la información por percepción del robot, toma los datos de entrada, los procesa, segmenta y decide, sin el surgimiento de esta rama no se conocería los métodos de procesos autónomos, sino solo se evidenciaría módulos básicos de procesamiento. (Reyes, 2015, pp.46-50)

1.3.1.1 Técnicas de visión artificial

Para el reconocimiento e interpretación de un dato con visión artificial se usa las técnicas de preprocesamiento, segmentación y extracción de características como se visualiza en la figura 8-1, en cada proceso las imágenes se convierten en una matriz de pixeles de información con una sucesión de métodos que generan la respuesta de salida deseada. (Reyes, 2015, pp.46-50)

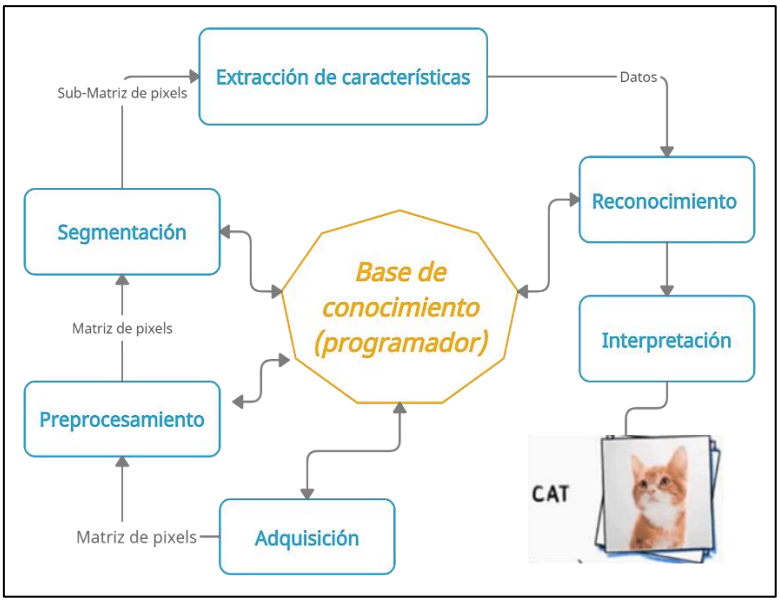


Figura 8-1: Proceso de un Sistema de Visión Artificial

Fuente: Reyes J., 2015.

Realizado por: Castañeda, Ruth V., 2021

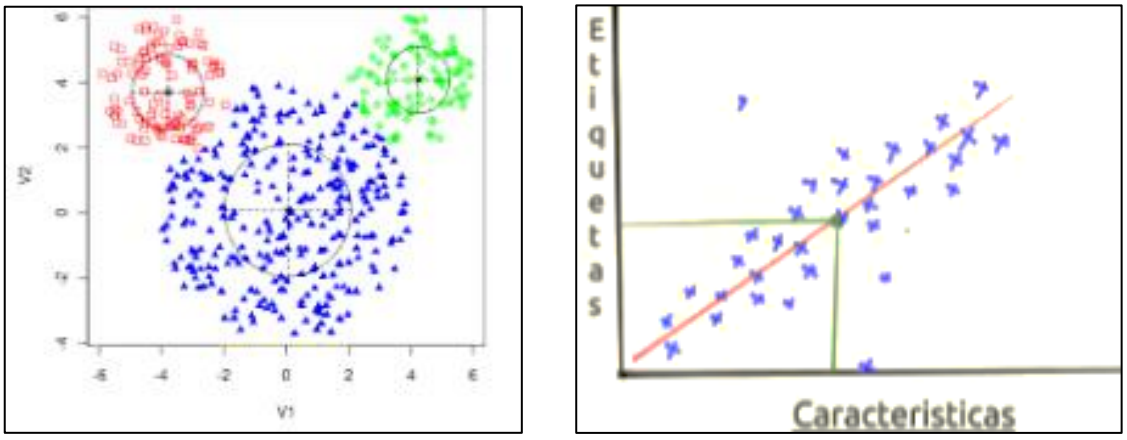
- **Adquisición:** Etapa para capturar la imagen en formato digital y procesarla en un software, las imágenes pueden ser adquiridas por una cámara RGB o ser tomadas de una secuencia de video. (Reyes, 2015, pp.46-50)
- **Pre-Procesamiento:** Una vez obtenida la imagen se usa técnicas de dominio en espacio del color para mejorar el brillo y apariencia de la matriz de datos, al eliminar el ruido previo a segmentarla. (Reyes, 2015, pp.46-50)
- **Segmentación*:*** Se enfoca en fraccionar el objeto para extraer las características, utilizando binarización, detección de bordes y momentos geométricos para determinar la ubicación del objeto a discriminar dentro de la imagen. (Reyes, 2015, pp.46-50)
- **Extracción de características:** Para tener los patrones de una imagen se puede hacer a través de un Aprendizaje Automático con Redes Neuronales Artificiales y manejar una arquitectura de datos en capas o segmentos. (Reyes, 2015, pp.46-50)
- **Reconocimiento:** Una vez extraídas las características se etiqueta los datos obtenidos para que la máquina comience a distinguir la información de salida y evaluar el comportamiento del sistema. (Reyes, 2015, pp.46-50)
- **Interpretación:** Se la conoce por la predicción que realiza al etiquetar las imágenes mostradas y verificar la efectividad que tiene en discriminar objetos en diferentes espacios. (Reyes, 2015, pp.46-50)

1.4 Aprendizaje Automático

El aprendizaje automático en la rama de la IA es un módulo capaz de resolver una problemática por medio de una estrategia aprendida, tiene un enfoque autónomo encargado de realizar tareas en base a las circunstancias de desarrollo, es un sistema capaz de adaptarse y crear repuestas con la experiencia acumulada, a medida que avanza aprovecha los errores y aprende de ellos, para mejorar la habilidad de los robots en toma de decisiones (Gimeno, 1994). Existen dos tipos de aprendizajes automáticos: supervisado y no supervisado, y se utilizan dependiendo del comportamiento de la predicción de salida.

1.4.1 Aprendizaje Supervisado

Es un algoritmo entrenado con patrones y etiquetas, en la cual segmenta un grupo de datos para extraer la información deseada y desarrollar un sistema de esquemas con módulos de aprendizaje en el procesador, existen dos algoritmos de aprendizaje supervisado: Clasificación y Regresión. La una ordena la información en categorías binarias y múltiples con el uso de patrones de categorización al comparar los datos de entrada en conjuntos de información como se visualiza en el literal a) figura 9-1, la siguiente determina un valor especifico a través de un gráfico de dispersión con un algoritmo de regresión lineal para conocer la predicción de salida como se visualiza en el literal b) figura 9-1. (Sandoval, 2015, pp.46-50).



(a) (b)

Figura 9-1: Aprendizaje Supervisado (a) Algoritmo de clasificación (b) Algoritmo de regresión
Fuente: Sandoval L., 2015

1.4.2 Aprendizaje no Supervisado

Es un algoritmo que no conoce las etiquetas de los datos de salida solo agrupa la información entre las características que comparten donde no existe una clase previa definida sino solo utiliza

métodos jerárquicos y particionales para formar niveles de segmentación con los módulos de grupos similares, este proceso se desarrolla como una interacción de datos no categorizados. (Sandoval, 2015, pp.46-50)

1.5 Red Neuronal Artificial

La RNA es un modelo basado en una red neuronal del cerebro formado por millones de neuronas compuestas por un Axón, un Soma y Dendritas, la información la trasmiten por estímulos eléctricos que se dan en las Dendritas, el Soma está a cargo de unir la información y el Axón de transmitir el mensaje final al cerebro. La RNA imita el proceso de las redes neuronales del cerebro usa un gran número de neuronas artificiales conectadas entre sí para que se agrupen en capas y determinar la salida con los valores de los pesos que componen la estructura neural. En la figura 10-1, se visualiza el funcionamiento de una Red Neuronal Artificial, las entradas multiplicadas por los pesos para pasar por una sumatoria a través de una función de transferencia y originar una neurona saliente. (Flores, 2020, pp.34-36)

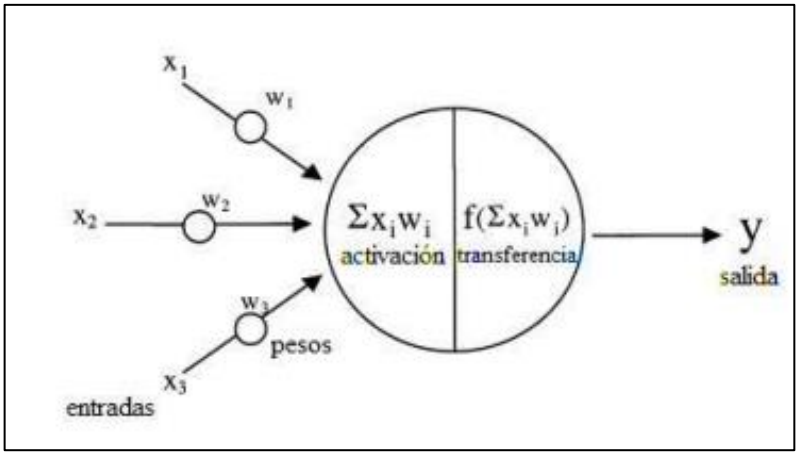


Figura 10-1: Modelo de una Red Neuronal Artificial

Fuente: Flores W., 2020

La figura 10-1 se expresa en forma gráfica la ecuación 1-1:

$$y_i = f_i\left(\sum_{j=1}^{n} w_{ij}\, x_j - \theta_i\right) \qquad (1.1)$$

Ecuación 1-1: Ecuación de una Red Neuronal Artificial

Donde:

i = nodos de la RNA

f_i = sección de activación y función de transferencia

$x_j =$ sección de entrada jth para cada nodo

$y_{i=}$ sección de salida de cada nodo

$w_{ij} =$ sección de pesos que tienen los nodos

$\theta_i =$ sección de sesgo de cada nodo

En la ecuación 1-1, se involucra las múltiples entradas x_j con los nodos de la estructura de activación para evaluar el manejo de los pesos en las conexiones y determinar el valor del sesgo en la neurona saliente. (Flores, 2020, pp.34-36)

Las redes neuronales trabajan a diferentes capas, por lo cual se clasifica según su red topológica en: Perceptrón Multicapa, Red de base radial, Red Neuronal Recurrente y Red Neuronal Convolucional.

1.5.1 Perceptrón Multicapa

Sé desarrolla a partir de una red neuronal monocapa compuesta por más de una capa intermedia oculta entre los datos de entrada y salida, puede tener una conexión de manera parcial o total dependiendo del umbral de datos a predecir, el valor del sesgo varía según el nivel de capas a procesar en la multiplicación de los pesos, con esto la neurona saliente toma la información recibida y codifica un valor para etiquetar los datos. (Calvo, 2017)

1.5.2 Red de Base Radial

La RBF es una red manejada por distancias entre capas, un punto de la arquitectura denominado centro, se encargan de realizar una distribución de transformación local sobre las señales de activación de las neuronas, el sistema de la red de base radial se maneja en valores de máximos y mínimos, donde los patrones que tengan un escaso recorrido para estar en el centro se los denomina salidas deseadas. (Longoni et al., 2010: p.696-697)

1.5.3 Red Neuronal Recurrente

La RNN es una estructura manejada en conexiones temporales entre las neuronas, a diferencia del perceptrón multicapa los datos de entrada se manejan en momentos con instantes de tiempos, para que el sistema de red tenga memoria y cree ciclos de trabajo en su arquitectura, con esta información la red aprende de manera recurrente de los procesos codificados en el sistema de programación. (Calvo, 2017)

1.5.4 Red Neuronal Convolucional

Una RNC es de aprendizaje supervisado, se encarga de extraer las características de una imagen a través de capas de conexiones con pesos compartidos, a diferencia de las otras redes neuronales esta trabaja con las dimensiones de entrada de las imágenes: ancho, alto y profundidad. En la figura 11-1 se visualiza el contraste que existe entre una Red por Perceptrón y una Red Convolucional, del lado izquierdo se mira las conexiones entrelazadas en vectores y a la derecha se tiene capas convolucionales en forma de bloques de imágenes. (Massiris et al., 2018: p.1023-1025)

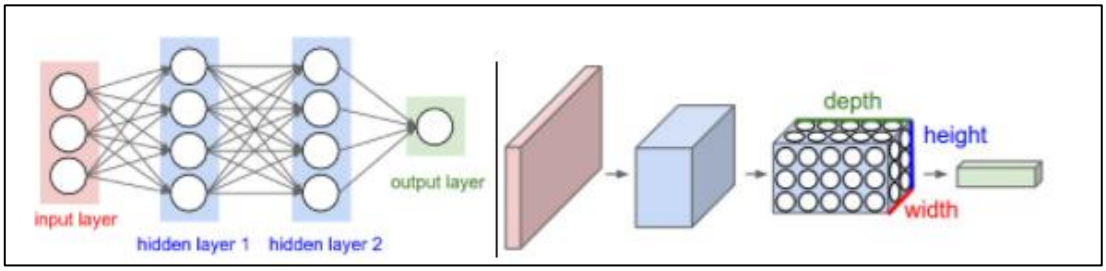


Figura 11-1: Red Neuronal Artificial vs Red Neuronal Convolucional
Fuente: Ibarra W., 2020

La RNA está compuesta por tres capas neuronales: capa de convolución, capa de agrupación y capa completamente conectada, todas ocupan un papel fundamental en el proceso de extracción de características y clasificación de objetos. (Ibarra, 2020, pp.34-36)

- Capa convolucional: se encarga de usar filtros y generar un producto escalar con las imágenes de entrada, las cuales recorren todas las neuronas entrantes en una matriz saliente con capas de neuronas ocultas, que contienen mapas de características.
- Capa de agrupación: esta capa enseña a las neuronas a reconocer un objeto en diferentes tipos de imágenes, para ello usa la variación espacial, detecta un objeto sin importar el ángulo que lo contenga, creando una agrupación de mapas de características. Existen tres tipos de agrupación: media, máxima y la suma.
- Capa completamente conectada*:* esta capa depende del proceso de las anteriores, donde toma los datos proporcionados y combina las características para crear una red neuronal convolucional que sea capaz de clasificar imágenes y emitir una predicción de los datos nuevos.

En la figura 12-1, se muestra los procesos de una red convolucional y como clasifica cada objeto detectado después de obtener una suma de mapas de características para predecir una imagen a partir de la arquitectura RNC.

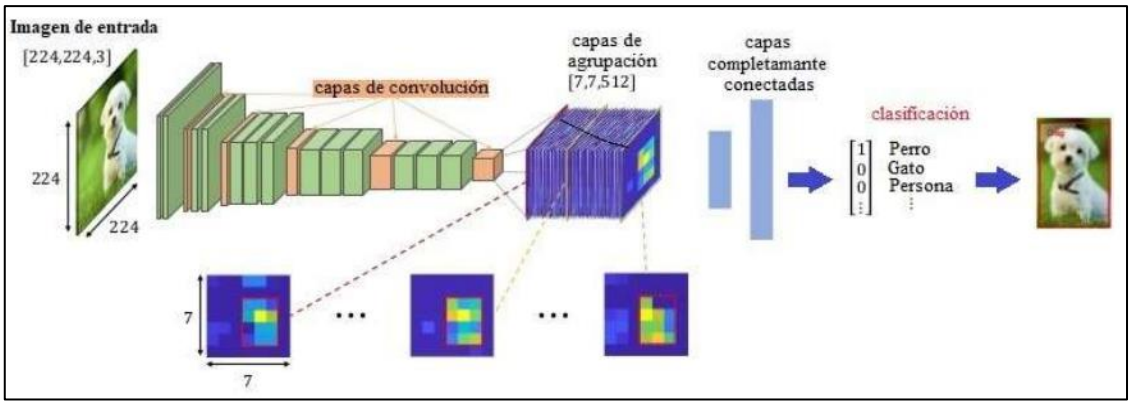


Figura 12-1: Arquitectura de una Red Neuronal Convolucional

Fuente: Ibarra W., 2020

CAPÍTULO II

2 MARCO METODOLÓGICO

En el presente capítulo se describen los requerimientos del hardware y software, donde a partir de esta sección al sistema se lo denomina ***Happy***, con la puntualización de estos parámetros se comienza a desarrollar la interfaz de enseñanza-aprendizaje.

- Requerimientos del Sistema El sistema disponga de una conexión inalámbrica Wifi.
- El Robot debe presentar al menos una autonomía de 30 minutos.
- El Robot permita desarrollar una interacción a través de voz, audio y video.
- El Robot tenga una configuración con un sistema de locomoción bípeda.
- El procesador del robot sea mínimo de 1.5Ghz de velocidad con un RAM de al menos 2GB.
- El Robot tenga una plataforma con entorno de desarrollo modular.
- Tener un robot con un sistema capaz de realizar algoritmos de *Machine Learning.*
- Que el robot se caracterice por medir menos de 60 cm de alto.
- Tener un robot con una estructura física amigable para el usuario.
- Que el robot sea de fácil transporte, operatividad y funcionalidad.

2.1

A continuación, se define los requerimientos para la implementación del sistema *Happy.*

2.2 Concepción general del Sistema

La concepción del sistema *Happy* se basa en tres subsistemas, detector de figuras: círculos, triángulos y cuadrados; detector de colores: amarillo, verde, azul y rojo; y detector de figuras con colores, para esta pedagogía de enseñanza-aprendizaje el docente debe elegir mediante un comando de voz el subsistema a trabajar, luego los estudiantes seleccionan una *flashcard* en base al parámetro definido y le exponen en la parte frontal al robot, o a su vez el docente puede hacerlo como se visualiza en la figura 1-2. El robot da la respuesta del sistema con el uso de un módulo de voz al identificar cada color, figura o figura con color, este método permite tener un juego interactivo con los niños para entrenar su memoria visual y conocimientos cognitivos a través de la experiencia.

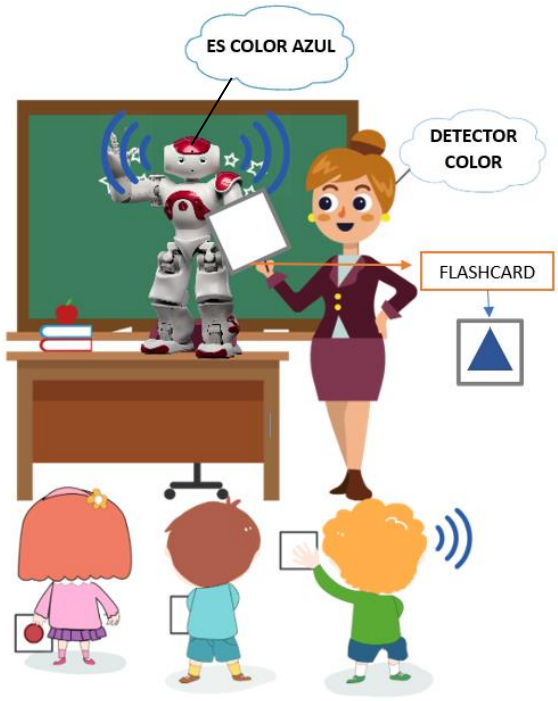


Figura 1-2: Concepción del Sistema

Realizado por: Castañeda, Ruth V., 2021

2.3 Arquitectura del Sistema

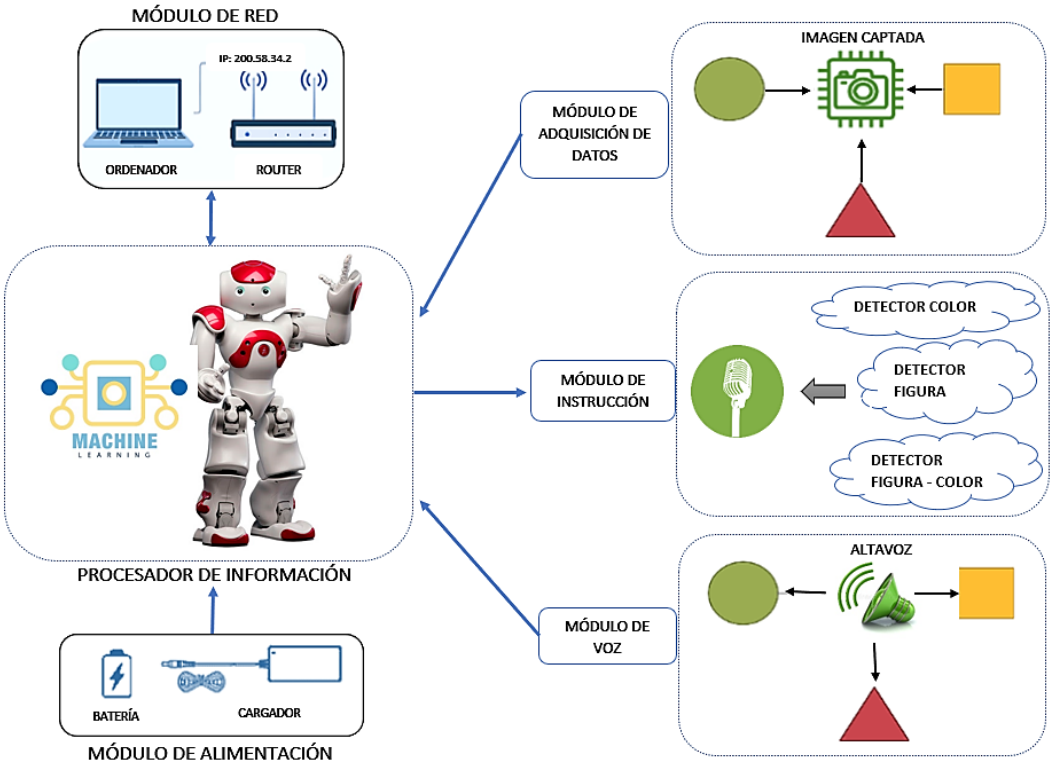


Figura 2-2: Arquitectura del Sistema

Realizado por: Castañeda, Ruth V., 2021

En la figura 2-2 se visualiza la arquitectura del sistema, en el módulo de alimentación se tiene el cargador con la batería en una autonomía mínima de 30 minutos, en el módulo de conexión se establece un enlace entre el ordenador y procesador a través de una red inalámbrica Wifi, en el módulo de adquisición de datos con la cámara interna del robot se captura la información entrante de las imágenes para ser procesadas, en el módulo de instrucción el micrófono recibe comandos de voz para determinar el subsistema a trabajar y en el módulo de voz con el uso del altavoz se emite la respuesta en base a la característica seleccionada.

2.4 Selección del hardware del Sistema Happy

2.4.1 Procesador de información

El procesador de información del sistema es el robot Nao versión V5, el cual cuenta con 25 grados de libertad para desplazarse de manera bípeda, 7 sensores táctiles distribuidos entre las extremidades inferiores, superiores y la cabeza, 4 micrófonos direccionales para interactuar con el usuario, un reconocimiento de voz para establecer un diálogo que puede ser en 20 idiomas diferentes, sin embargo, el idioma del sistema *Happy* es el castellano, dos cámaras 2D usadas en reconocimiento de formas, objetos y personas, su estructura es de policarbonato en un diseño blanco con azul. En la figura 3-2 se visualiza donde se encuentran los componentes del robot Nao y el diseño de su arquitectura. (SOFTBANK, 2018)

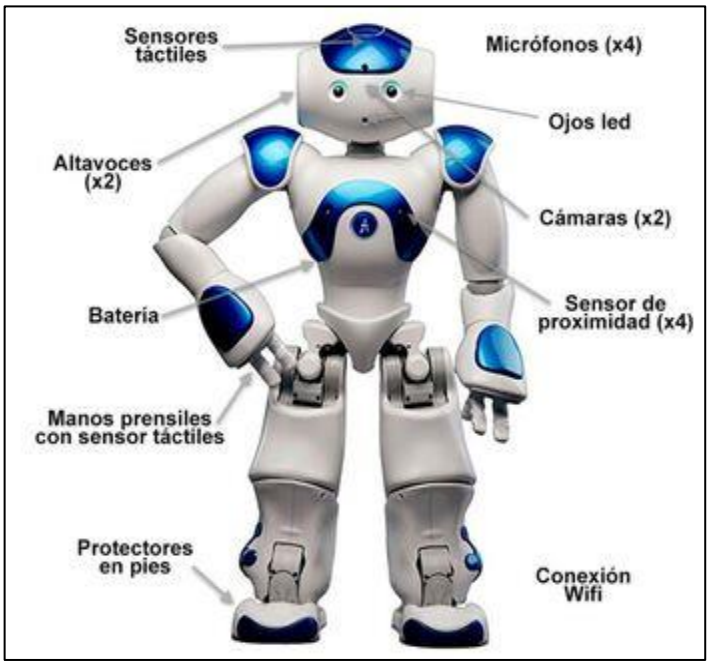


Figura 3-2: Distribución de los componentes del robot Nao

Fuente: SOFTBANK, 2018

Las especificaciones técnicas del procesador se detallan en la tabla 1-2, con una velocidad de reloj superior a la requerida y con una memoria RAM de 4GB, la cual es suficiente para el sistema. (SOFTBANK, 2018)

Tabla 1-2: Especificaciones del Procesador del robot Nao

Procesador	UPC	Velocidad de reloj	RAM
Atom E3845	Cuatro núcleos	1,91 GHz	DDR3 de 4 GB

Fuente: SOFTBANK, 2018
Realizado por: Castañeda, Ruth V., 2021

2.4.2 Módulo de Conexión

En el módulo de conexión se identifica la dirección IP que tiene el *Router* en el *Kid* del robot Nao, al presionar una vez el botón del pecho comienza a mencionar la dirección de su conexión de enlace en el campo Host/IP, después en el ordenador se configura las propiedades TCP/IPv4 en base a la red del robot, con esto se tiene todos los dispositivos en una misma red para acceder a la plataforma de desarrollo, otro de las opciones para ingresar es a través de un cable ethernet, si en caso la conexión inalámbrica falla se configura la red interna con el programa *ROBOT SETTINGS*, en la figura 4-2 se visualiza la configuración del módulo. (SOFTBANK, 2018)

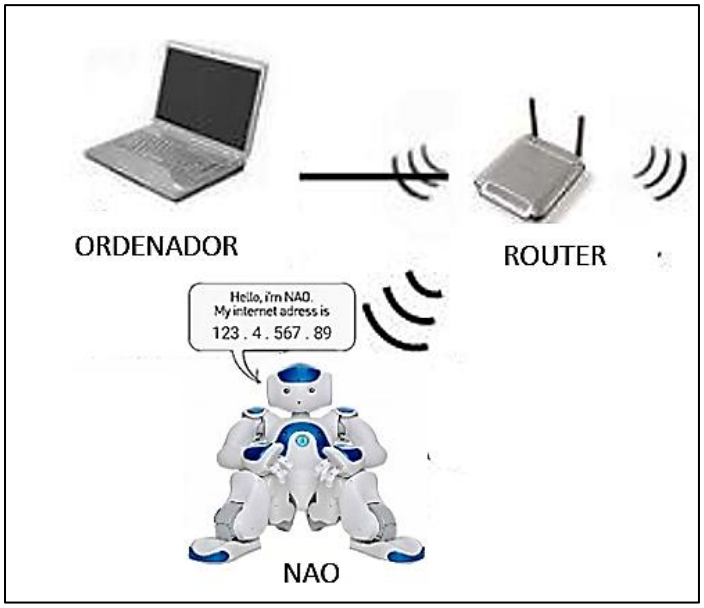


Figura 4-2: Módulo de Conexión
Fuente: SOFTBANK, 2018

2.4.3 Módulo de Alimentación

En el módulo de alimentación se tiene una batería con una duración de 90 minutos cargada en su totalidad según el tiempo de uso y manejo, en la tabla 2-2 se detalla la distribución de los Iones de Litio de la batería en su capacidad nominal de voltaje y corriente. (SOFTBANK, 2018)

Tabla 2-2: Características técnicas de la Batería del robot Nao

Tipo	Iones de Litio
Voltaje / capacidad nominal	21,6 V / 2,9 Ah
Voltaje de carga máximo	25,2 V
Corriente de carga recomendada	1.8 ~ 2. 1A
Corriente máxima de carga / descarga	2.1A / 2.0A
Energía	62,5 Wh
Duración de carga	90 min

Fuente: SOFTBANK, 2018
Realizado por: Castañeda, Ruth V., 2021

2.4.4 Módulo de Adquisición de datos

El módulo de adquisición de datos se compone por la cámara interna del robot Nao que tiene la capacidad de ajustar la imagen a 6 resoluciones de pixeles en base al valor del ID como se detalla en la tabla 3-2, los datos de sensibilidad, producción, y vista de las descripciones técnicas de la cámara se presentan en la tabla 4-2. Las librerías *Naoqi* para manejar los parámetros de visión son: *AlPhotoCapture* que toma fotografías, *ALVideoDevice* se encarga de administrar las entradas de video, *ALColorBlobDetection* divisa manchas de un color determinado, y por último *ALVisionRecognition* la cual hace que el robot detecte, aprenda y reconozca patrones de una imagen. (SOFTBANK, 2018)

Tabla 3-2: Resolución de imagen de la cámara del robot Nao con el valor del ID

Nombre de ID de parámetro	Valor de ID	Descripción
AL :: KQQQQVGA	8	Imagen de 40 * 30px
AL :: KQQQVGA	7	Imagen de 80 * 60px
AL :: KQQVGA	0	Imagen de 160 * 120px
AL :: KQVGA	1	Imagen de 320 * 240px
AL :: KVGA	2	Imagen de 640 * 480px
AL :: K4VGA	3	Imagen de 1280 * 960px

Fuente: SOFTBANK, 2018
Realizado por: Castañeda, Ruth V., 2021

Tabla 4-2: Características de la Matriz de imagen, Sensibilidad, Producción y Vista

Cámara	Modelo	MT9M114
	Tipo	Sensor de imagen CMOS
Matriz	Resolución	1,22 Mp
	Formato óptico	1/6 pulgada
	Píxeles activos (HxV)	1288x968px
Sensibilidad	Tamaño de píxel	1,9 µm * 1,9 µm
	Gama dinámica	70 dB
	Relación señal / ruido (máx.)	37 dB
	Responsividad	2,24 V / lux-seg (550 nm)
Producción	Salida de cámara	1280 * 960 a 30 fps
	Formato de datos	(Espacio de color YUV422)
	Tipo de persiana	Persiana enrollable electrónica
Vista	Campo de visión	72,6 ° DFOV (60,9 ° HFOV, 47,6 ° VFOV)
	Rango de enfoque	30cm ~ infinito
	Tipo de enfoque	Foco fijo

Fuente: SOFTBANK, 2018
Realizado por: Castañeda, Ruth V., 2021

En la figura 5-2, se visualiza la ubicación de las cámaras 2D del robot Nao y la representación angular de su proyección visual, en el eje horizontal tiene un ángulo de 60,97° y en el eje vertical de 47,64° con una diferencia de 39,7° entre la cámara de la cabeza y mentón. (SOFTBANK, 2018)

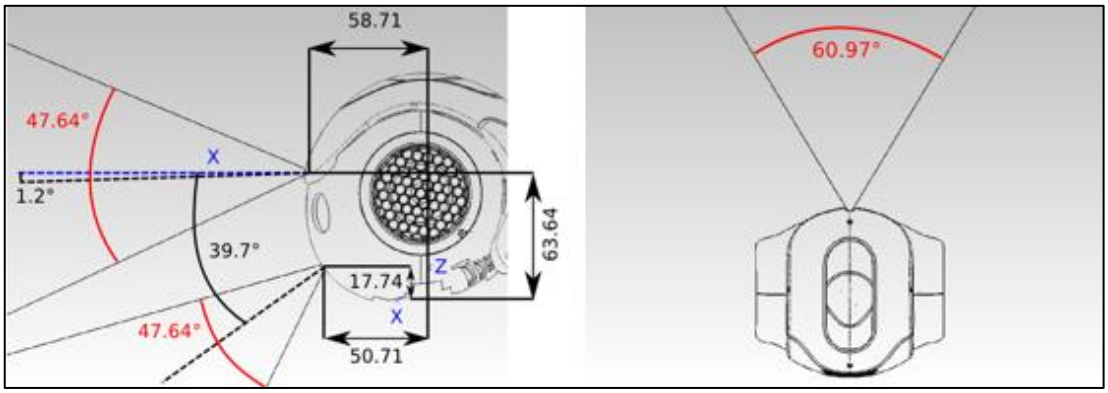


Figura 5-2: Ángulo de visión de las cámaras 2D del robot Nao
Fuente: SOFTBANK, 2018

2.4.5 Módulo de Instrucción

El módulo de instrucción se compone de 4 micrófonos en la cabeza del robot Nao, las cuales tienen especificaciones técnicas de sensibilidad y frecuencia como se detalla en la tabla 5-2, hace uso de librerías *Naoqi* como: *ALSpeechRecognition* y *ALVoiceEmotionAnalysis*, la una se encarga

de comprender las preguntas del usuario y la siguiente identifica la emoción expresada en el hablante. (SOFTBANK, 2018)

Tabla 5-2: Especificaciones técnicas del Micrófono de Nao, sensibilidad y frecuencia.

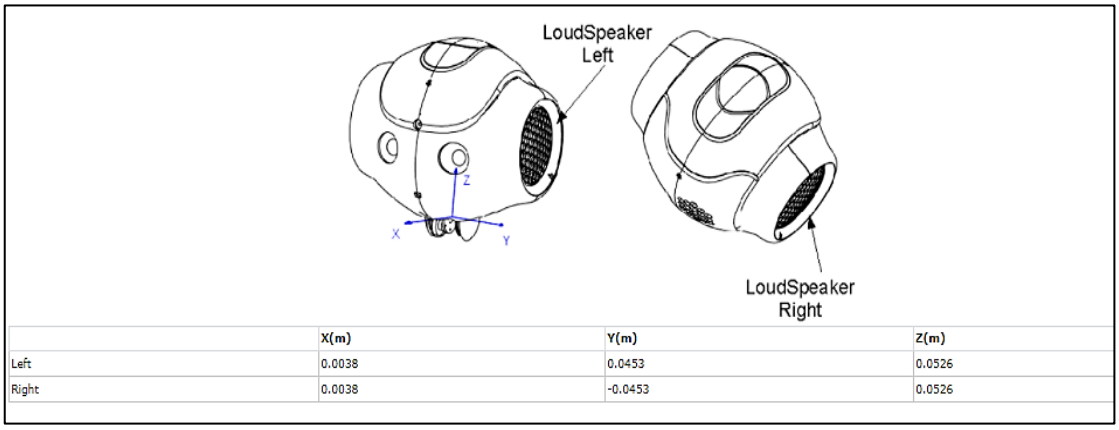
Micrófonos	*x*4 – cabeza
Sensibilidad	20mV/Pa +/- 3dB a 1KHz
Rango de frecuencia	150Hz a 12kHz

Fuente: SOFTBANK, 2018

Realizado por: Castañeda, Ruth V., 2021

2.4.6 Módulo de voz

El módulo de voz está compuesto por altavoces en los lados laterales de la cabeza tipo estéreo como se visualiza en la figura 6-2, su sistema de voz se maneja con librerías *Naoqi* que sintetizan la voz a través de *AlTextToSpeech* este asigna el idioma, la velocidad y la entonación de la voz al modular la respuesta del audio en la información de salida. (SOFTBANK, 2018)



	X(m)	Y(m)	Z(m)
Left	0.0038	0.0453	0.0526
Right	0.0038	-0.0453	0.0526

Figura 6-2: Ubicación de los parlantes del robot Nao

Fuente: SOFTBANK, 2018

2.4.7 Ordenador

El sistema *Happy* maneja módulos de Machine Learning con Visión Artificial, por lo cual se usa un ordenador portátil con un procesador *Intel Core i7*, una memoria RAM de 8GB con un Disco duro de 750GB y el sistema operativo Windows 10 para crear una comunicación bidireccional con el robot Nao ver figura 7-2, estas características técnicas mejoran la respuesta en la codificación de los procesos de segmentación y entrenamiento, un ordenador inferior a estas

especificaciones técnicas toma más tiempo en compilación, debido al esfuerzo que requiere el sistema.

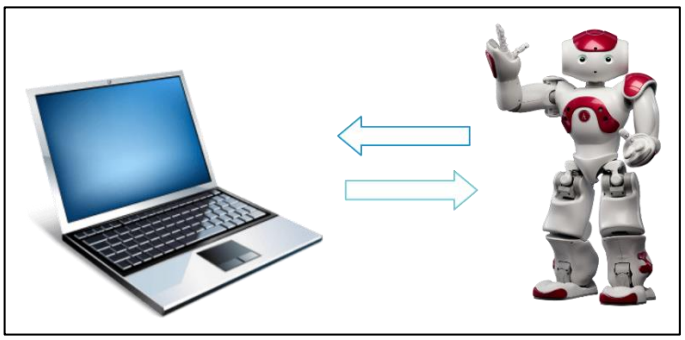


Figura 7-2: Ordenador portátil con el robot Nao

Realizado por: Castañeda, Ruth V., 2021

2.4.8 Flashcards del Sistema

Las *flashcards* del sistema *Happy*, están en formato A4(21x29,7cm) con una imagen en la sección central de un cuadrado, un círculo, o un triángulo distribuidas en colores primarios, se usa imágenes sin etiqueta y sin leyenda con pigmentos de color laser como se visualiza en la figura 8-2, para las pruebas se usa figuras con diferentes tamaños, pero con las mismas características.

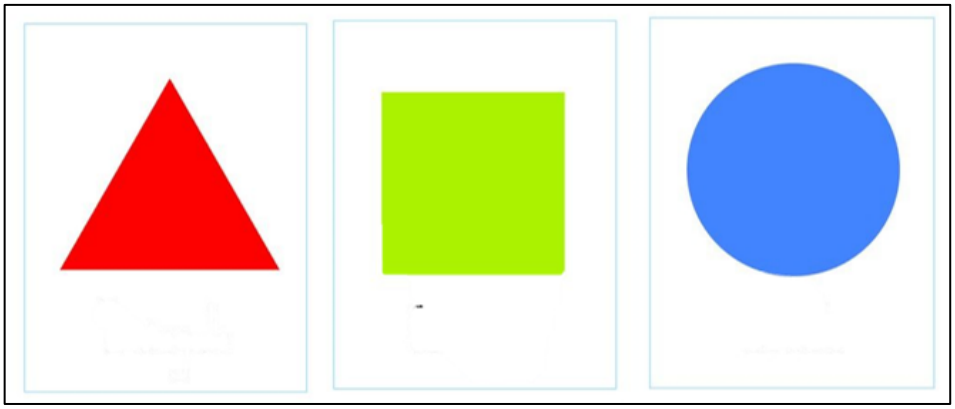


Figura 8-2: Flashcards

Realizado por: Castañeda, Ruth V., 2021

2.5 Selección del software del Sistema Happy

El procesador de información del sistema tiene 4 plataformas de edición de código y una interfaz de programación en bloques denominada *Choregraphe* que permite acceder de manera grafica a los módulos del robot, en la tabla 6-2 se detalla la compatibilidad que existe entre estos softwares

y las funciones del robot Nao, para identificar que plataforma es accesible en el manejo de técnicas de visión artificial. (SOFTBANK, 2018)

Tabla 6-2: Comparación de los tipos de Plataformas de desarrollo en el robot Nao

Software de programación	**Aplicación en ejecución**		***Choregraphe***	
	Ordenador	Robot Nao	Apps	Edición de código
Python	✓	✓	✓	✓
C++	✓	✓	*x*	*x*
Java	✓	*x*	*x*	*x*
JavaScript	✓	✓	✓	*x*

Fuente: SOFTBANK, 2018

Realizado por: Castañeda, Ruth V., 2021

2.5.1 Choregraphe

Choregraphe es un lenguaje de desarrollo en bloques de fácil manejo e intuitivo, es propio del robot Nao y de la empresa *Softbank Robotics*, este programa da acceso a los módulos del humanoide para crear rutinas de movimientos, aplicaciones audiovisuales y reconocimiento de objetos, cada bloque tiene internamente un editor predefinido de Python como se visualiza en la figura 9-2. (SOFTBANK, 2018)

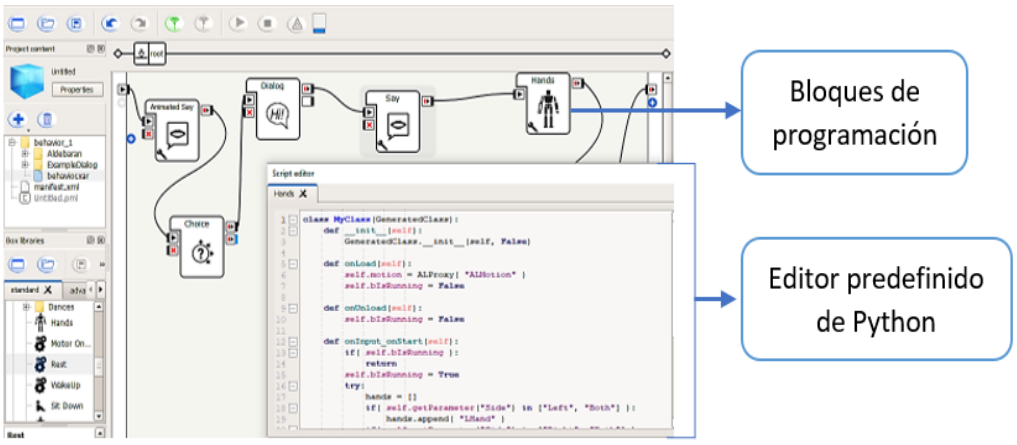


Figura 9-2: Interfaz de programación Choregraphe

Realizada por: Castañeda, Ruth V., 2021

2.5.2 Python 2.7

Python es un software de código abierto con una multiplataforma capaz manejar grandes volúmenes de datos, las librerías que tiene el sistema dan acceso a crear módulos predictivos con algoritmos de aprendizaje automático que siguen instrucciones en un lenguaje funcional. Este software tiene un intérprete que verifica la sintaxis al compilar, luego lo traduce en un código Byte para pasarlo al MVP o Máquina Virtual Python y como último ejecutar la programación, ver figura 10-2 (Robledano, 2021). Para programar el robot Nao, es necesario trabajar con la versión 2.7 de Python, debido a que el *SDK-Naoqi* no soporta versiones posteriores, de igual manera solo es compatible con librerías de esta versión.

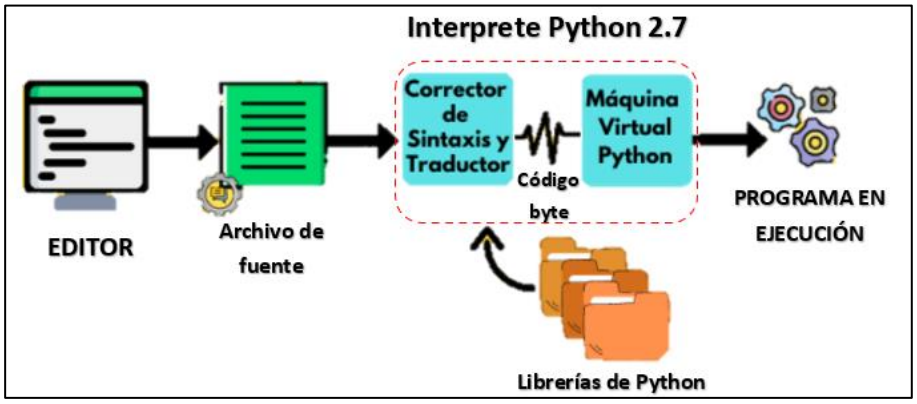


Figura 10-2: Proceso de Compilación Python 2.7

Fuente: Robledano, 2021

Realizada por: Castañeda, Ruth V., 2021

2.5.3 SDK Naoqi

Naoqi es el software encargado de controlar las librerías del robot Nao, antes de programar en una plataforma se debe crear un enlace con el *SDK Naoqi* de lo contrario no es posible acceder al procesador, el uso de este SDK vincula la comunicación con las diferentes librerías de movimiento, voz, audio y video. En la figura 11-2, se detalla el nombre de las librerías del robot con el módulo que están vinculados. (Aldebaran, 2017)

En la programación de Nao con Python 2.7, se debe instalar el SDK *Naoqi* en el ordenador y configurar las variables de entorno para acceder a las librerías de los módulos, si este enlace no se genera no se puede controlar las funciones del robot.

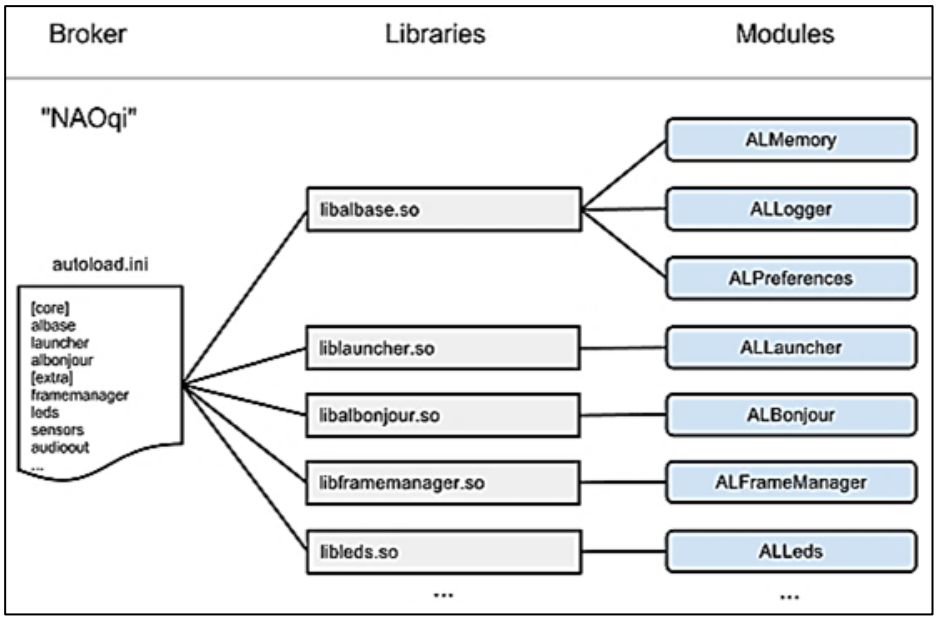


Figura 11-2: Árbol de módulos de NAOqi

Fuente: ALDEBARAN, 2017

2.5.4 PyCharm

PyCharm es un IDE multiplataforma en la programación de Python, se caracteriza por tener un fácil acceso a las librerías con su interfaz de edición de código inteligente, el desarrollador antes de ejecutar el código final da paso a identificar los errores sin necesidad de compilar, la versión que se usa es *PyCharm Community Edition* 2020.3.33 que es compatible con la versión Python 2.7, su interfaz se visualiza en la figura 12-2. (JETBRAINS, 2021)

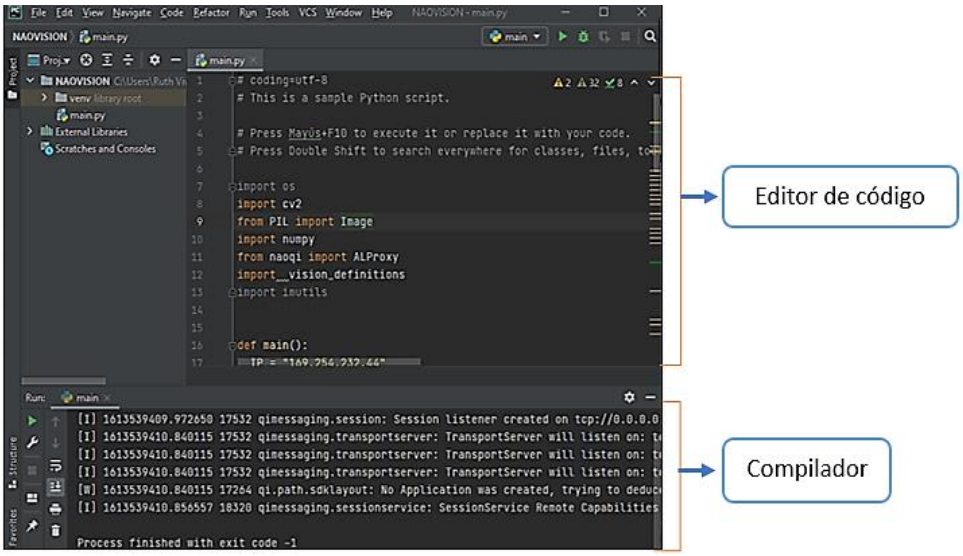


Figura 12-2: Interfaz de programación de Python con PyCharm

Realizado por: Castañeda, Ruth V., 2021

2.5.5 Módulo de entrenamiento OpenCV

OpenCV es un módulo de Visión por Computador de código abierto para realizar sistemas de Visión Artificial, una de las ventajas que tiene es permitir crear algoritmos de una imagen y ejecutarla en un bloque de información, los parámetros para procesar pueden realizar operaciones e implementar técnicas de visión en diferentes muestras de datos (Marín, 2020). La versión que se utiliza es *OpenCV* 3.0.0 que es compatible con Python 2.7 con las librerías *Numpy* y *Matplotlib* para crear una matriz de información con las imágenes de entrada y procesar los datos, en la figura 13-2 se visualiza el Logo de *OpenCV* con Python.

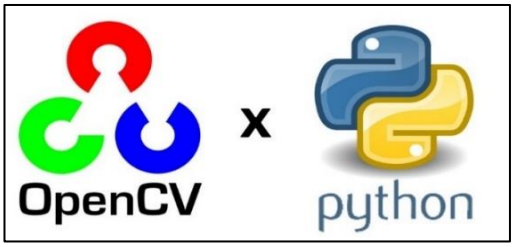


Figura 13-2: Logo de OpenCV con Python

Fuente: Marín, 2020

2.5.5.1 Interfaz gráfica Cascade Trainer GUI

El software *Cascade Trainer GUI* es un entrenador de datos con clasificadores *HAAR*, con *OpenCV* se usa desde la versión 3.0 para analizar imágenes, es una plataforma que permite modificar parámetros como: el número de etapas, relación de aceptación, pre calcular el valor con el tamaño del índice y el número de hilos, ver figura 14-2, de estos depende el tiempo que toma en procesar las imágenes ingresadas para el entrenamiento. (AMIN, 2017)

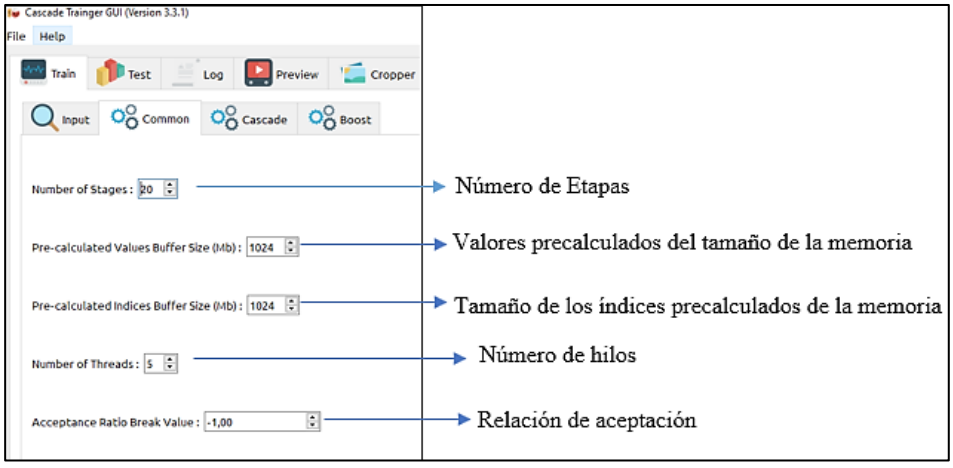


Figura 14-2: Interfaz Cascade Trainer GUI

Realizado por: Castañeda, Ruth V., 2021

2.5.5.2 Clasificador Haar Cascade

Haar Cascade es un tipo de clasificador en cascada para entrenar imágenes en *OpenCV*, toma una gran cantidad de datos positivos y negativos para crear un mapa de tipologías en dos bloques de información. Cada bloque tiene un valor único que se obtiene al restar la suma de los píxeles del rectángulo blanco y la suma de los píxeles del rectángulo negro como se visualiza en la figura 15-2, cuando finaliza el entrenamiento de las imágenes se adquiere un archivo de *Lenguaje Marcado Extensible (XML)*, para verificar si el archivo detecta se llama a esta extensión con librerías *OpenCV* dentro del código del sistema. (DOXYGEN, 2018)

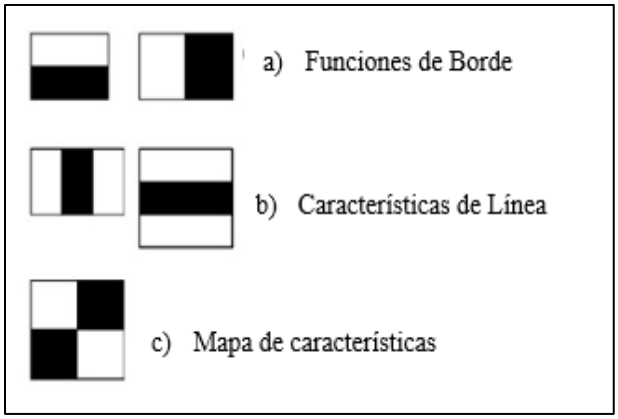


Figura 15-2: Función Haar en una imagen
Fuente: DOXYGEN, 2018

2.6 Desarrollo del algoritmo del Sistema Happy

Para implementar el sistema *Happy* se sigue una secuencia de procesos, las cuales se detallan a continuación:

2.6.1 Adquisición de Datos

En la adquisición de datos se toma la cámara del mentón del robot Nao con un ID=1 y una resolución de 320*240px a 30fps, luego en la plataforma Python se incluye la librería *camProxy* que está inmerso en el paquete *Naoqi* para capturar las imágenes como se detalla en el Anexo A. En la figura 16-2 se verifica la resolución de la cámara del robot Nao al capturar las *flashcards* en una distancia de 57cm, esta información se comienza a receptar y guardar en el procesador para los siguientes procesos.

Figura 16-2: Imagen capturada por el robot Nao con una resolución de 320*240px a 30fps

Realizado por: Castañeda, Ruth V., 2021

2.6.2 Preprocesamiento

En el preprocesamiento la imagen capturada es mejorada antes de segmentarla y para eso se usa los espacios de color, una herramienta que analiza cada píxel que compone la imagen y los convierte en una matriz distribuida de información, existen diferentes modelos en espacio de color, entre ellos el modelo RGB y HSV, el cual matemáticamente extrae las particularidades de cada pigmentación de una imagen.

2.6.2.1 Modelo RGB

Es un modelo de color que usan las cámaras para construir una imagen, está basada en tres intensidades de luz como son el rojo, verde y azul, donde sumadas entre si con diferentes valores pueden crear nuevos pigmentos. En la figura 17-2, se visualiza la representación del espacio RGB, donde a través de un cubo unitario con ejes se distingue las coordenadas de cada color para discriminar los pigmentos, el valor de los parámetros permite identificar con valores numéricos el color de la figura para generar etiquetas correctas. (Aguirre, 2013, pp.42-45)

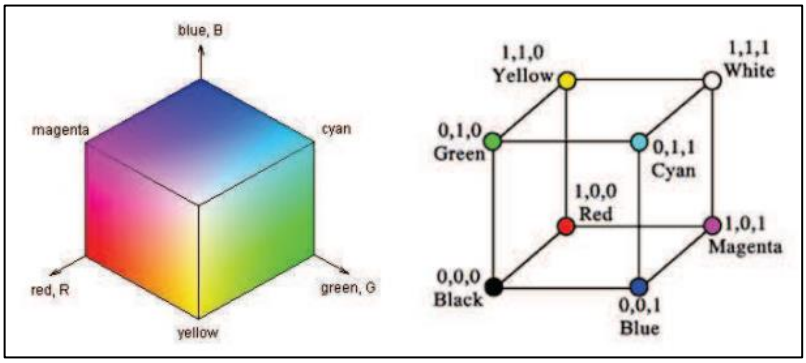


Figura 17-2: Espacio de color del modelo RGB

Fuente: Aguirre N., 2013

2.6.2.2 Modelo HSV

El modelo de color HSV se particulariza en tres aspectos de la imagen que son en el tono, saturación y brillo, cada uno representa un espacio geométrico en el cono de base *quasi-hexagonal* como se muestra en la figura 18-2. Los tres aspectos del espacio de color HSV están definidas en tres magnitudes que van en rango de valores para puntualizar un color. (Aguirre, 2013, pp.42-45)

Las magnitudes están compuestas por los siguientes valores:

- H (tono): El rango es de 0-360°, la gama cromática está definida por una rueda circular para expresar su posición.
- S (saturación): Los valores están de 0-100% dependiendo de la cantidad de pigmento.
- B (brillo): Los valores están igual de 0-100% en base a la luminosidad.

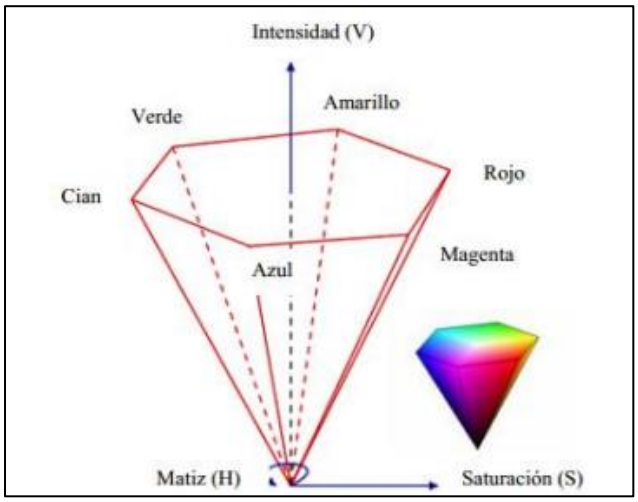
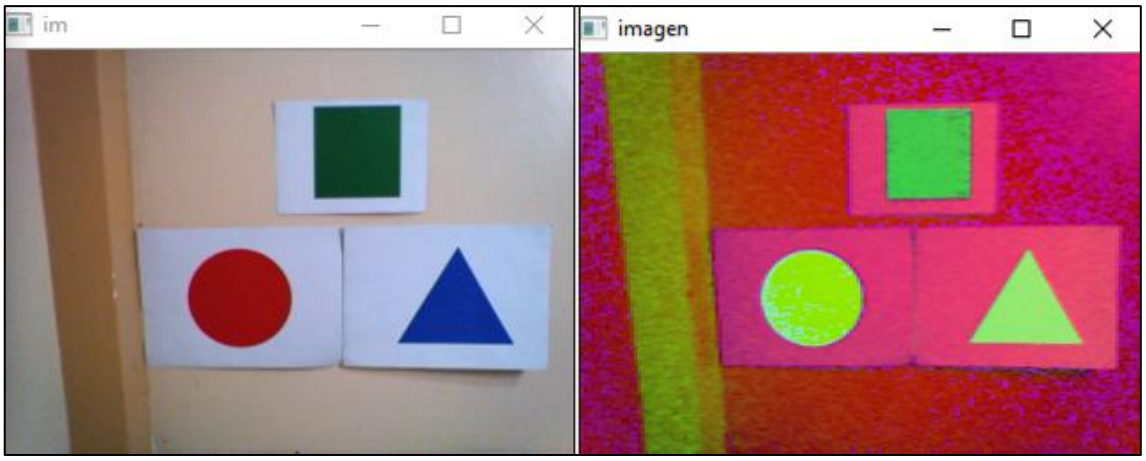


Figura 18-2: Cono Quasi-hexagonal del modelo HSV
Fuente: Aguirre N., 2013

Después que la imagen es capturada por Nao, se hace un preprocesamiento como se visualiza en la figura 19-2 para luego segmentarla, en el literal (a) se tiene un espacio de color en RGB con datos de entrantes y estás se convierten en espacio HSV como esta en el literal (b), para extraer la saturación de la imagen, la gama del color y el brillo en base al cono de medida *quasi-hexagonal*



(a) (b)

Figura 19-2: (a) Imagen en espacio RGB, (b) Imagen en espacio HSV

Realizado por: Castañeda, Ruth V., 2021

2.6.3 Segmentación

La segmentación se encarga de subdividir una imagen digital en clases y subconjuntos para diferenciar el segmento en el cual se extrae las características a procesar, en la cual con la umbralización o procesamiento por *Threshold* (umbral) se separa objetos del fondo dependiendo los niveles de gris y texturas que tenga la imagen, es una técnica no paramétrica basada en regiones a partir del valor del umbral. (Ramírez, 2017)

2.6.3.1 Método de Umbralización

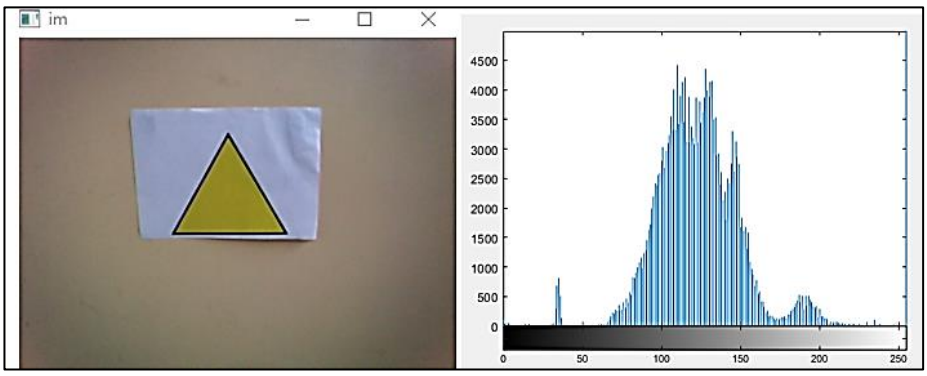
El Método de umbralización segmenta un objeto de una imagen, a través del histograma, el valor del *Threshold*, la detección de bordes y eliminación de ruido, con estas características se limita la información de los datos de entrada. (Ramírez, 2017)

2.6.3.1.1 Histograma

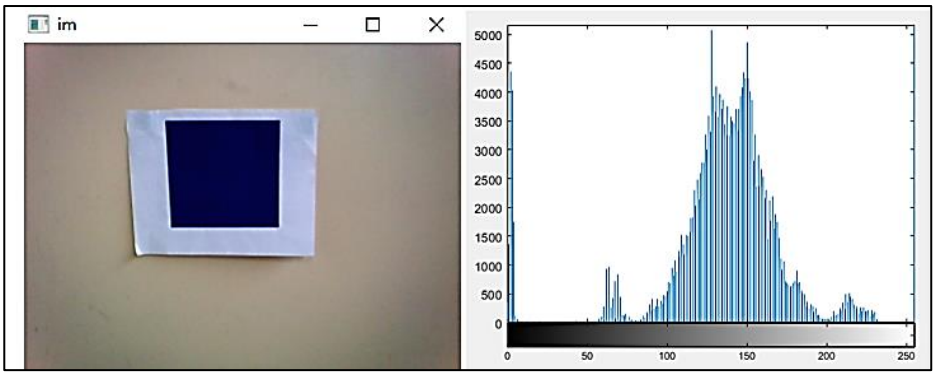
El valor del umbral de una imagen se determina con un histograma, para obtener los datos numéricos a través de una representación gráfica en barras e identificar como están distribuidas las cifras y variables de cada píxel dentro de la imagen. (Fernández, 2019. pp.28-30)

En la figura 20-2, se visualiza el histograma de cada gráfico de los *flashcards*, esto se obtiene por medio de las librerías de *OpenCV*. En el literal (a) se muestra el triángulo de color amarillo con una distribución de pixeles que va desde 0 a 4500 con una intensidad de luz 60 a 205; en el literal (b) la distribución del cuadrado azul llega a 5000 pixeles con una intensidad de 60 a 235; y en el

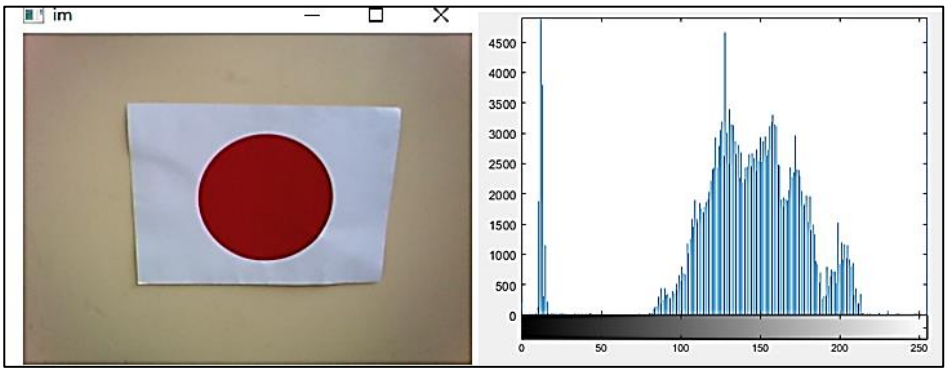
literal (c) la distribución del círculo rojo va hasta 4600 píxeles con una intensidad de 85 a 210, con estos valores se conoce la segmentación de pigmentos en las figuras.



(a)



(b)



(c)

Figura 20-2: Histograma de las figuras geométricas (a) Triangulo (b) Cuadrado (c) Círculo.
Realizado por: Castañeda, Ruth V., 2021

2.6.3.1.2 Método OTSU

El método *OTSU* es un tipo de umbralización global, busca una forma estadística para determinar el valor del umbral en una imagen usando la varianza como una medida de dispersión. Para

determinar el valor adecuado del umbral en una imagen se hace a través de una sucesión de pasos. (Fernández, 2019. pp.28-30)

- ***Paso 1:*** Realizar un histograma de la imagen para obtener la distribución gráfica de la distribución de datos.
- ***Paso 2:*** Tener sumas acumulativas de los datos obtenidos de la imagen a partir del histograma con los siguientes parámetros:
  - $f(x, y) = imagen\ MxN$
  - $\{0,1,2, \dots, L-1\} = niveles\ de\ saturación$
  - $n_i = número\ de\ pixeles\ con\ intensidad$
  - $p_i = \frac{n_i}{MN} = Probabilidad\ de\ que\ ocurra\ el\ nivel\ de\ intensidad$

$$P_1(k) = \sum_{i=0}^{k} p_i \ = Sumas\ acumulativas\ donde\ P_1(k), k = 0,1, \dots, L-1 \qquad (1.2)$$

Ecuación 1-2: Ecuación de la Suma de Probabilidades de Niveles de Intensidad

La ecuación 1-2 establece la probabilidad de que determinado píxel pertenezca a una clase especifica.

$C_1 = conjunto\ de\ píxeles\ con\ intensidad\ de\ [0, k]$

$C_2 = conjunto\ de\ píxeles\ con\ intensidad\ de\ [k+1, L-1]$

- ***Paso 3:*** Media acumulativa $m(k)$, ver ecuación 2-2.

$$m(k), k = 0,1, \dots, L-1$$

$$m(k) = \sum_{i=0}^{k} ip_i \qquad (2.2)$$

Ecuación 2-2: Ecuación de la Media Acumulativa

- ***Paso 4:*** Media global m_g, ver ecuación 3-2

$$m_g = \sum_{i=0}^{L-1} ip_i \qquad (3.2)$$

Ecuación 3-2: Ecuación de la Media Global

- ***Paso 5:*** Varianza entre clases ver ecuación 4-2, se encarga de medir el grado de intensidad entre dos conjuntos de clases, para discriminar el valor del umbral.

$$\sigma_{B^2}(k) = \frac{[m_G P_1(K) - m(K)]^2}{P_1(K)[1 - P_1(K)]} \qquad (4.2)$$

Ecuación 4-2: Ecuación de la Varianza entre clases

- ***Paso 6:*** Maximizar el valor de la varianza obtenido a través de la ecuación 5-2:

$$k^* = \max_{0 \le k \le L-1} \sigma_{B^2}(k) \qquad (5.2)$$

Ecuación 5-2: Ecuación del valor máximo de la Varianza

- ***Paso 7:*** Como último paso se obtiene la medida de separabilidad a partir del cociente entre la varianza de las clases y la global. ver ecuación 6-2.

$$\mu^* = \frac{\sigma_{B^2}(k)}{\sigma_{G^2}} \qquad (6.2)$$

Ecuación 6-2: Ecuación de la Medida de Separabilidad

A continuación, en la figura 21-2, se visualiza las imágenes binarizadas capturadas por la cámara del robot Nao, a través de librería *cv2.threshold* que se encuentra dentro paquete de *OpenCv*, se tiene un círculo, un cuadrado y un triángulo que son las figuras geométricas usadas en las *flashcards*, en este proceso solo se segmenta el objeto analizar dejando a un lado cualquier elemento que este dentro de la imagen.

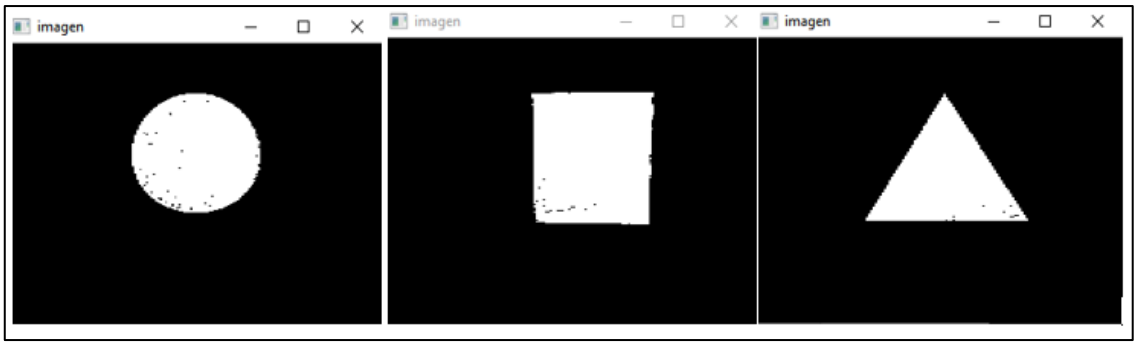


Figura 21-2: Binarización de las figuras geométricas a través de la cámara del robot Nao
Realizado por: Castañeda, Ruth V., 2021

2.6.3.1.3 Eliminación de Ruido

Una imagen cuando es binarizada siempre tienen un porcentaje mínimo de ruido, pero en ocasiones para segmentarla puede traer dificultades, por lo tanto, se busca tener una máscara limpia para procesarla. En Python con el uso de OpenCV se eliminó el ruido de las imágenes a través del comando de programación *cv2.morphologyEx* y *cv2.MORPH_OPEN.*

2.6.3.1.4 Detección de bordes

Una vez que a la imagen se la tiene binarizada y eliminado el ruido, se hace una detección de borde para encontrar la franja de aire que se separa la muestra con los demás objetos. Este proceso

busca precisión al momento de separar el elemento principal, a continuación, en la figura 22-2, se visualiza la imagen de las figuras geométricas con su detección de borde a través del uso de la librería *cv2.findcontours* en Python como se detalla en el Anexo B, el color del borde de cada elemento cambia al depender del tono de la figura.

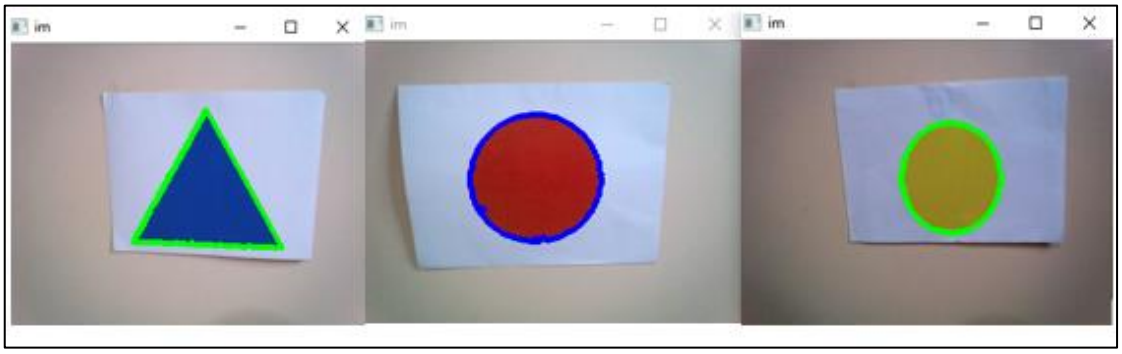


Figura 22-2: Detección de los bordes a las figuras geométricas, con la cámara de Nao

Realizado por: Castañeda, Ruth V., 2021

Una vez determinados los contornos de cada figura, se calculó los centros a partir de momentos geométricos. Matemáticamente a los momentos se los define en la ecuación 7-2:

$$M_{ij} = \sum_x \sum_y x^i y^i I(x,y) \qquad (7.2)$$

Ecuación 7-2: Ecuación del Momento Geométrico

En la ecuación 7-2, *X* y *Y* son las coordenadas de un determinado píxel, indicando su grado de intensidad. Para determinar los centros hay tres momentos que es necesario conocer y son M00, M01 Y M10, estos valores se reemplazan en la fórmula de momento de la siguiente manera:

$$M_{00} = \sum_x \sum_y x^0 y^0 I(x,y) = \sum_x \sum_y I(x,y)$$

$$M_{01} = \sum_x \sum_y x^0 y^1 I(x,y) = \sum_x \sum_y y I(x,y)$$

$$M_{10} = \sum_x \sum_y x^1 y^0 I(x,y) = \sum_x \sum_y x I(x,y)$$

Una vez obtenido cada momento, se calculó cada coordenada en ***x*** y ***y***, para el valor de ***x*** se divide M10/M00 y para ***y*** M01/M00, obteniendo lo siguiente:

$$\frac{M10}{M00} = \frac{\sum_x \sum_y x I(x,y)}{\sum_x \sum_y I(x,y)} = \frac{x1 + x2 + \cdots + xn}{n} = \bar{x}$$

$$\frac{M01}{M00} = \frac{\sum_x \sum_y yI(x,y)}{\sum_x \sum_y I(x,y)} = \frac{y1 + y2 + \cdots + yn}{n} = \bar{y}$$

En la figura 23-2, se visualiza los centros de las imágenes a partir de la aplicación de momentos geométricos para tener las coordenadas ***x*** y ***y,*** con estos valores se determina el centro del triángulo, del cuadrado y del círculo, en el Anexo C se detalla el módulo de programación de esta sección.

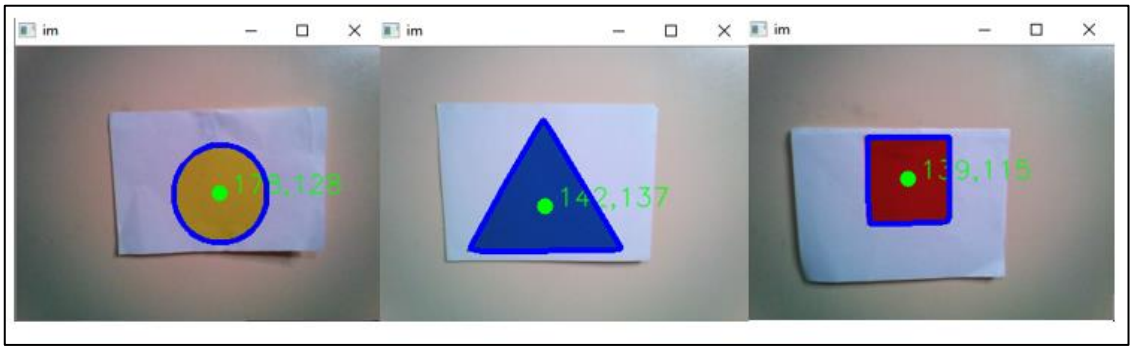


Figura 23-2: Detección de los centros de las figuras geométricas, con la cámara de Nao

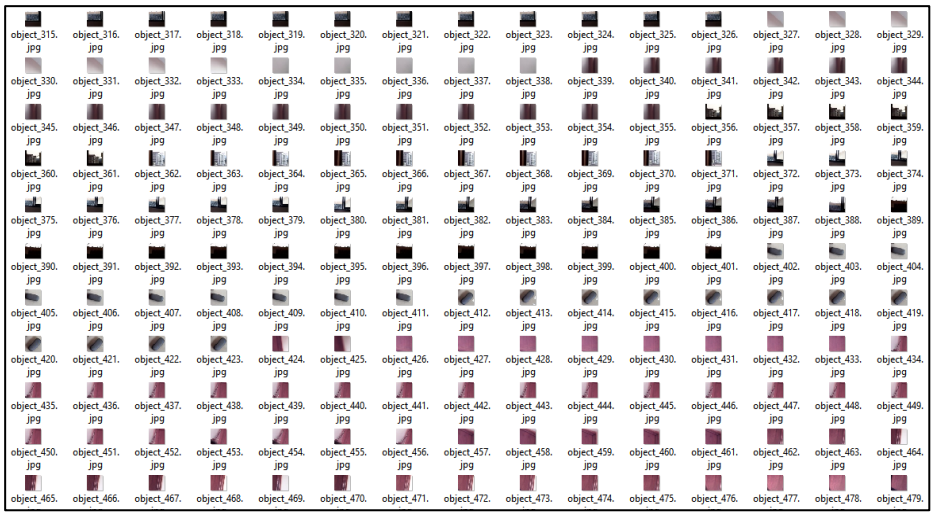
Realizado por: Castañeda, Ruth V., 2021

2.6.4 Extracción de características

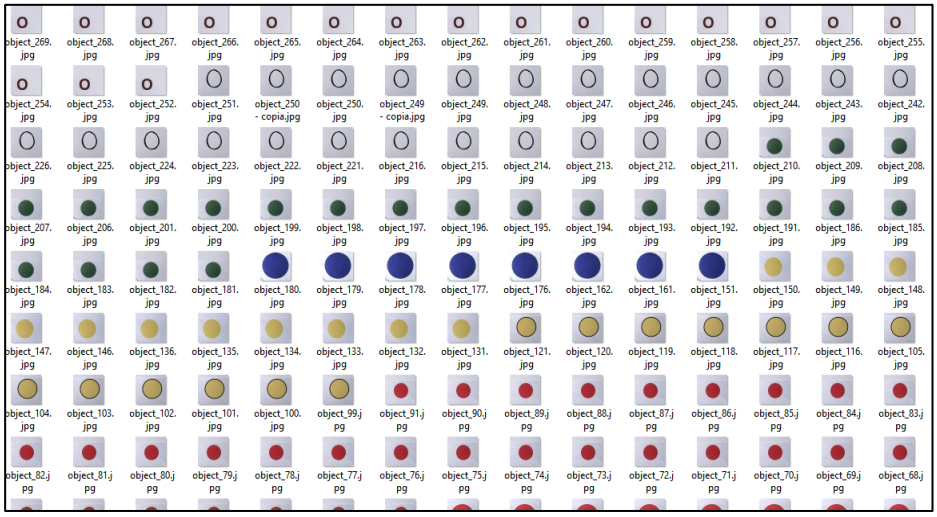
Para extraer las características de una imagen se usa un algoritmo de aprendizaje automático supervisado con un módulo de desarrollo capaz de entrenar datos a través de patrones y particularidades escondidas, una vez terminado el proceso el sistema es capaz de clasificar varios tipos de información etiquetadas, donde las RNA actúan para crear una estructura de entrenamiento mediante distribuciones de redes neuronales.

2.6.4.1 Estructura de Entrenamiento

En la estructura de entrenamiento se maneja la plataforma *Cascade-Trainer-GUI* con un clasificador tipo HAAR y un núcleo de Red Neuronal Convolucional, en la cual se capturan 300 imágenes positivas y 500 negativas de 38x36 píxeles, con 1368 RNC en tres canales RGB ver figura 24-2, estas se almacenan en dos carpetas una con la letra **P** y otra con la letra **N** para que al momento de procesar reconozca la etiqueta señalada, cuando finaliza el entrenamiento se entrega un archivo *.XML* para implementarlo en el módulo predictivo de las figuras geométricas como se detalla en el Anexo F. Se entrena tres tipos de figuras: cuadrados, triángulos y círculos, para ejecutar tres archivos *.XML* y cargarlos en al sistema *Happy*, el robot Nao cuando tiene el algoritmo entrenado en su memoria es capaz de reconocer el tipo de figura que visualiza y las menciona con el módulo de voz *ALTextToSpeech*.



(a)



(b)

Figura 24-2: Imágenes en Cascade-Trainer-GUI (a) Negativas
(b) Positivas

Realizado por: Castañeda, Ruth V., 2021

2.6.4.2 Algoritmo de Aprendizaje Supervisado

Al obtener el módulo entrenado a través de redes neuronales convolucionales en un archivo *XML*, se carga al sistema *Happy* de Python para clasificar los círculos, triángulos y cuadrados como están en el código del Anexo D, después se importa esta información al robot Nao con los archivos de extensión entrenados, luego procede a predecir la figura geométrica que está dentro del *flashcard* de salida.

En la figura 25-2 se describe a través de una diagramad de flujo el proceso del sistema *Happy* para detectar figuras.

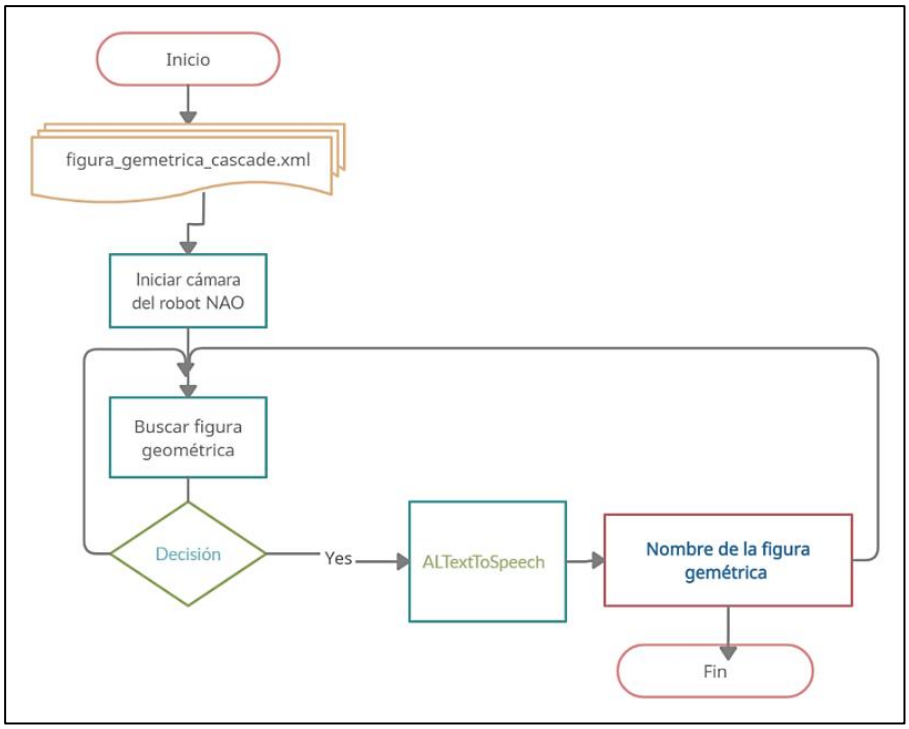


Figura 25-2: Diagrama de flujo del Sistema *Happy*, para detector de figuras.

Realizado por: Castañeda, Ruth V., 2021

A continuación, se presenta los pasos del Sistema *Happy* en base al diagrama de flujo de la figura 23-3:

1. Se importan las librerías usadas en Python *naoqi*, *numpy*, *matplotlib*, *cv2*, *PIL y visión_difinitions* ver anexo A.
2. Se inician las variables globales del sistema.
3. Se cargan los archivos entrenados con una extensión de archivo *XML*.
4. Se inicia la cámara del mentón de Nao con el id=1.
5. Se realiza el acondicionamiento de la imagen detectada.
6. Se enmarca la figura detectada y se etiqueta la figura.
7. Con la librería *ALTextToSpeech*, sé menciona el nombre de la figura.
8. Una vez mencionada la figura regresa a buscar una nueva.

2.7 Herramienta de evaluación del Algoritmo de Aprendizaje

Para evaluar un algoritmo entrenado se usa la matriz de confusión encargada verificar como se está comportando el sistema al clasificar figuras, en el campo de la inteligencia artificial esta matriz visualiza el desempeño de un algoritmo para predecir las etiquetas de los datos de salida. (Barrios, 2019)

2.7.1 Matriz de Confusión

La matriz de confusión es un conjunto de parámetros métricos para evaluar la efectividad de un algoritmo de aprendizaje, en la tabla 7-2 se muestra la distribución de valores reales y predichos de un sistema.

Tabla 7-2: Matriz de confusión, parámetros del modelo entrenado

MATRIZ DE CONFUSIÓN (MC)		**Clasificador**	
		Positivos	Negativos
Valor Real	Positivos	VP	FN
	Negativos	FP	VN

Fuente: Barrios J., 2019

Realizado por: Castañeda, Ruth V., 2021

Donde:

VP: El modelo entrenado clasifica correcto los positivos.

VN: El modelo entrenado clasifica correcto los negativos.

FN: El modelo entrenado clasifica a los positivos como negativos.

FP: El modelo entrenado clasifica a los negativos como positivos.

La matriz de confusión tiene métricas para identificar cómo se comporta el modelo, las cuales son:

Exactitud: analiza todos los casos positivos, para determinar lo cerca que están los resultados del valor verdadero, ver ecuación 8-2. (Barrios, 2019)

$$exactitud = \frac{VP+VN}{VP+FP+FN+VN} \qquad (8.2)$$

Ecuación 8-2: Ecuación de exactitud con respecto a la MC

Precisión: determina la cantidad de predicciones fueron correctas, tanto positivas como negativas con respecto al total, ver ecuación 9-2. (Barrios, 2019)

$$precisión = \frac{VP}{VP+FP} \qquad (9.2)$$

Ecuación 9-2: Ecuación de precisión con respecto a la MC

Sensibilidad: se la denomina tasa de verdaderos positivos, se encarga de analizar los casos positivos que el algoritmo ha clasificado correctamente, ver ecuación 10-2. (Barrios, 2019)

$$sensibilidad = \frac{VP}{VP+FN} \qquad (10.2)$$

Ecuación 10-2: Ecuación de sensibilidad con respecto a la MC

Especificidad: se la denomina tasa de verdaderos negativos, se encarga de analizar como el algoritmo ha clasificado los casos que son negativos, ver ecuación 11-2. (Barrios, 2019)

$$especificidad = \frac{VN}{VN+FP} \qquad (11.2)$$

Ecuación 11-2: Ecuación de especificidad con respecto a la MC

CAPÍTULO III

3 PRUEBAS Y RESULTADOS

En esta sección se analizan los resultados del sistema *Happy*, el cual se dividió en tres subsistemas, detector de colores: amarillo, verde, rojo y azul; detector de figuras geométricas: cuadrado, triángulo y círculo; y por último detector del color con el tipo de figura. Para evaluar los parámetros de distancia de cada imagen con respecto al robot, se hace pruebas en entornos controlados y no controlados.

3.1 Detector de colores

La prueba consistió en tomar 22 imágenes de cada color en entornos controlados y 13 no controlados a magnitudes de 20cm, 40cm, 60cm y 80cm, para verificar la distancia adecuada de trabajo entre las *flashcards* con el robot Nao e identificar que tonalidad de color le es fácil detectar al sistema *Happy*.

3.1.1 Entornos Controlados

Esta prueba se realizó en espacios donde el robot Nao no tuvo perturbaciones externas y con buena iluminación, las *flashcards* que se mostraron eran dirigidas a la cámara del mentón en magnitudes proporcionales a 20cm, 40cm, 60cm y 80cm como se visualiza en la figura 1-3.

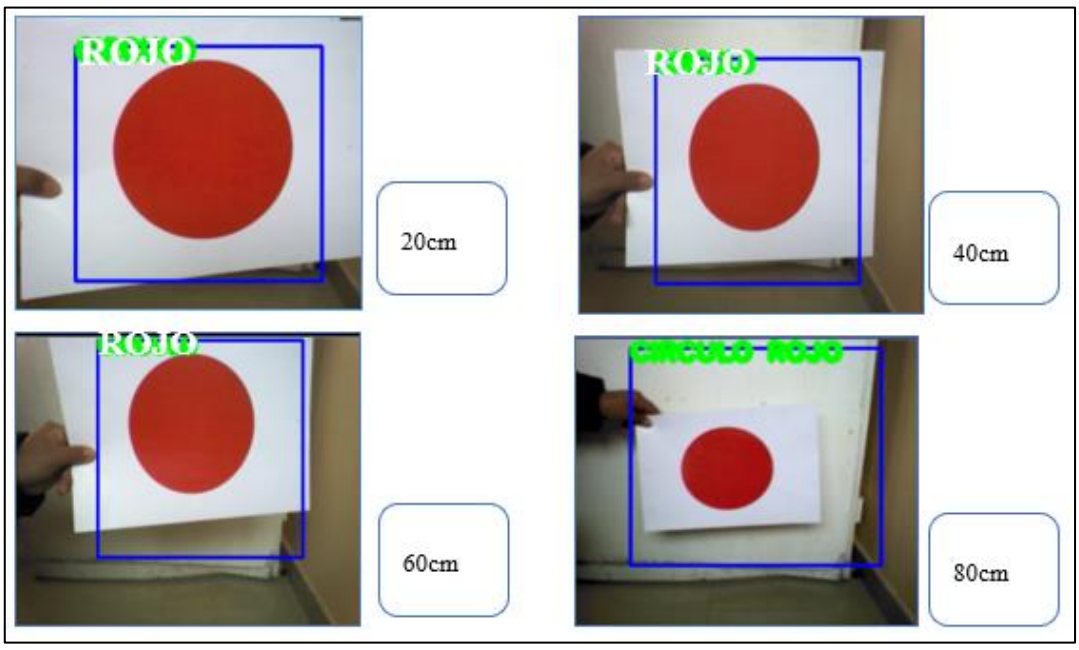


Figura 1-3: Detector de colores – Entornos Controlados

Realizado por: Castañeda, Ruth V., 2021

En la tabla 1-3, se detalla el porcentaje de aciertos que tuvo el detector de colores a diferentes distancias con la muestra de 22 *flashcards* de cada color y un total de 88 imágenes, luego se identificó el color que tuvo mayor acierto en todas las distancias y la magnitud adecuada de trabajo.

Tabla 1-3: Tabla porcentual del Detector de Colores – Entornos Controlados

DETECCIÓN DE COLORES	Distancias			
	20cm	40cm	60cm	80cm
Verde	81,81%	95,45%	90,90%	86,36%
Amarillo	81,81%	90,90%	90,90%	81,81%
Azul	95,45%	95,45%	90,90%	86,36%
Rojo	95,45%	95,45%	90,90%	86,36%

Realizado por: Castañeda, Ruth V., 2021

En el gráfico 1-3, se detalla a través de una modelo de barras el comportamiento del detector de colores en las distancias mencionadas, en base a los resultados obtenidos se determina que el sistema en ambientes controlados se debe manejar en una distancia de 40 a 60cm para tener buenos resultados con el módulo de adquisición de imágenes, donde los colores que tienen mejor percepción son rojo y azul con un porcentaje de 95,45%.

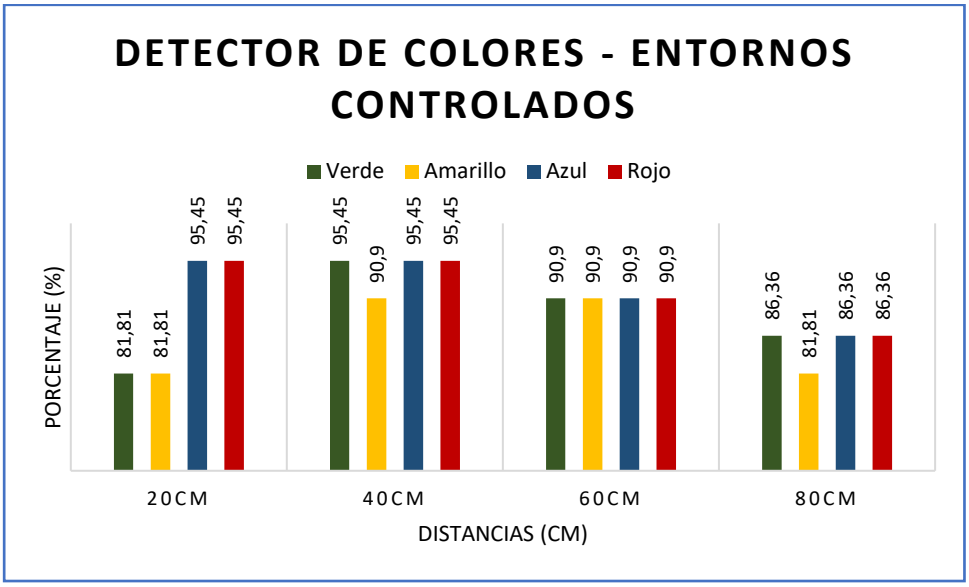


Gráfico 1-3: Modelo en barras del Detector de Colores - Entornos Controlados

Realizado por: Castañeda, Ruth V., 2021

3.1.2 Entornos no Controlados

La prueba se realizó en el Centro de Educación Inicial "San Pablo" en la ciudad de Riobamba en clases presenciales con 38 estudiantes divididos en tres grupos, el robot Nao tuvo que adaptarse al espacio normal de trabajo de los niños, los factores de perturbación externa fueron: el tipo de luz, movimiento constante al mostrar las *flashcards* e interferencia de audio, la distancia que se manejo fue aproximada a 20cm, 40cm, 60cm y 80cm ver figura 2-3. La metodología de enseñanza-aprendizaje fue un tipo de juego interactivo, el cual consistió en darles una *flashcard* a cada niño para que se acerquen delante del robot con su tarjeta, y verificar como actúa el detector de colores del sistema.

Figura 2-3: Detector de colores – Entornos no Controlados

Realizado por: Castañeda, Ruth V., 2021

En la tabla 2-3, se detalla el porcentaje de aciertos que tiene el detector de colores en entornos no controlados a diferentes distancias con una muestra de 13 *flashcards* por cada color, debido a la emergencia sanitaria se tuvo un tiempo máximo de 20 minutos por clase y se manejó 52 *flashcards*. Al realizar la didáctica se tomó las medidas sanitarias pertinentes con los niños, en la cual se tuvo requisito para estar en la clase el uso de mascarilla y el distanciamiento dentro del espacio de trabajo.

Tabla 2-3: Tabla porcentual del Detector de Colores – Entornos no Controlados

DETECCIÓN DE COLORES	Distancias			
	20cm	40cm	60cm	80cm
Verde	76,90%	84,61%	84,61%	76,90%
Amarillo	76,90%	84,61%	76,90%	69,20%
Azul	84,61%	92,30%	92,30%	84,61%
Rojo	92,30%	92,30%	92,30%	84,61%

Realizado por: Castañeda, Ruth V., 2021

En base al grafico 2-3 y al análisis anterior en ambientes controlados se determina que el sistema *Happy* se debe manejar en un rango de distancia de 40cm a 60cm para que el módulo de adquisición de datos y de voz tenga buenos resultados, no obstante, en ambientes no controlados el porcentaje en aciertos descendió con respecto a la prueba anterior de un 95,45% a 92,30%, por lo cual se recomienda generar un ambiente controlado para trabajar con el sistema.

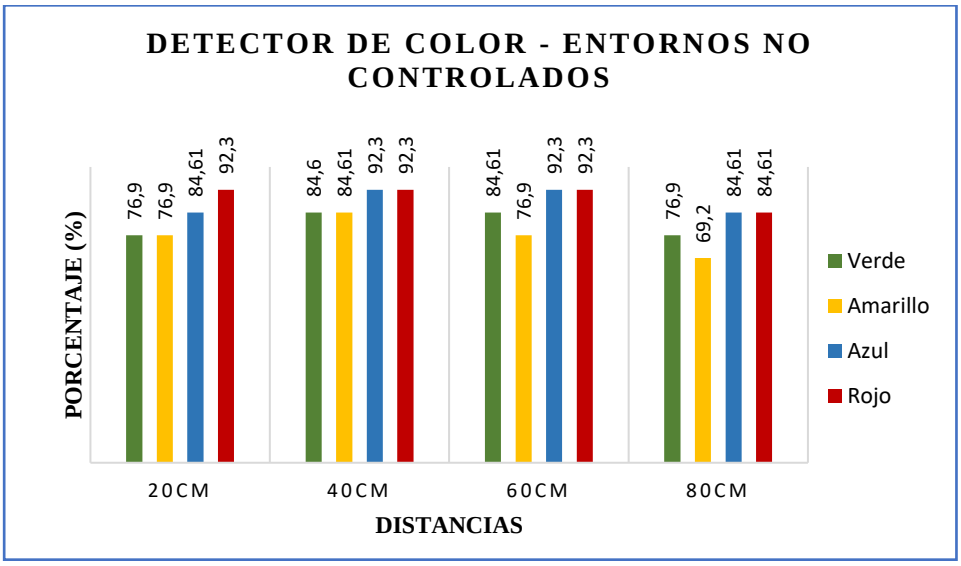
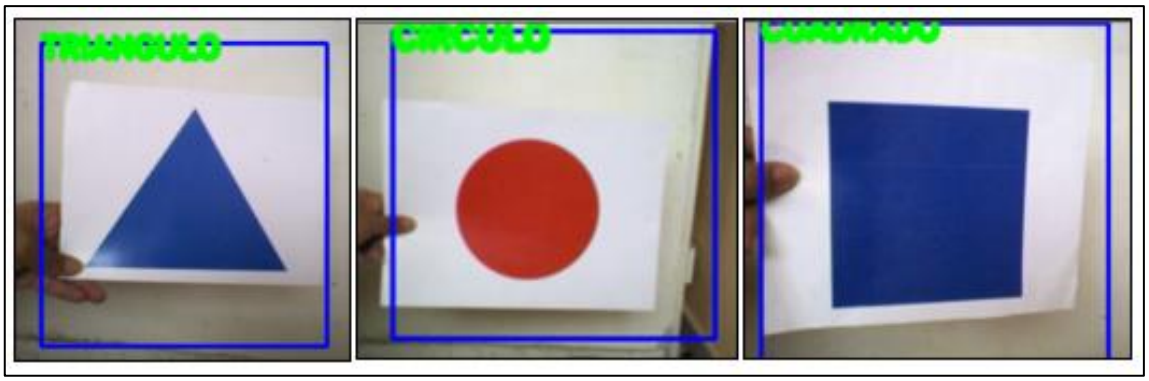


Gráfico 2-3: Modelo en barras del Detector de colores – Entornos no Controlados

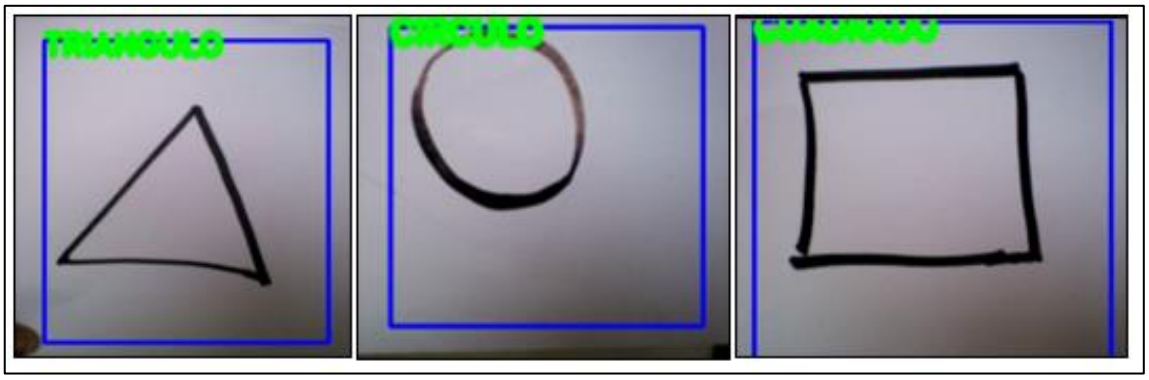
Realizado por: Castañeda, Ruth V., 2021

3.2 Detector de Figuras Geométricas

Para evaluar el detector de figuras geométricas se usó la Matriz de Confusión en cada modelo entrenado: círculos, triángulos y cuadrados. En esta prueba se usó imágenes que el sistema *Happy* ha visto e imágenes nunca vistas como el de un círculo dibujado con marcador ver literal b) de la figura 3-3, en base al resultado de las pruebas anteriores se determina al rango de distancia a trabajar de 40cm a 60cm.



(a)



(b)

Figura 3-3: Detector de círculos por el robot Nao (a) Imagen vista (b) Imagen nunca vista
Realizado por: Castañeda, Ruth V., 2021

3.2.1 Detector de Cuadrados

Después de determinar que el algoritmo pudo detectar figuras nunca vistas se procedió hacer pruebas con las *flashcards* como se visualiza en la figura 3-3. En la tabla 3-3 se detalla los aciertos de falsos positivos, falsos negativos, verdaderos negativos y verdaderos positivos que tuvo el Sistema *Happy* con el modelo entrenado del detector de cuadrados, la muestra usada fue de 40 imágenes, donde 25 correspondían a cuadrados y 15 eran indefinidas.

Tabla 3-3: Tabla de la Matriz de Confusión del Detector de Cuadrados

MATRIZ DE CONFUSIÓN FIGURA CUADRADA		**Clasificador**	
		Positivos	Negativos
Valor Real	Positivos	23	2
	Negativos	1	14

Realizado por: Castañeda, Ruth V., 2021

Una vez obtenido los datos se calculó las métricas de exactitud, precisión, sensibilidad y especificidad del módulo predictivo.

$$exactitud = \frac{23+14}{23+14+1+2} = 0.925x100 = 92.5\% \quad (8.2)$$

$$precisión = \frac{23}{23+1} = 0.95x100 = 95\% \quad (9.2)$$

$$sensibilidad = \frac{19}{19+2} = 0.904 = 90{,}4\% \quad (10.2)$$

$$especificidad = \frac{14}{14+1} = 0.933x100 = 93{,}3\% \quad (11.2)$$

Con estos datos se determina que el modelo entrenado al tener una especificidad de 93,3% su detección de falsos positivos es inferior a los demás parámetros, además tiene una precisión de 95% dando como resultado un buen modelo de aprendizaje.

3.2.2 Detector de Círculos

En la tabla 4-3 se visualiza la matriz de confusión del detector de círculos, la muestra usada es 40 *flashcards*, 25 figuras correctas y 15 elementos indistintos, de la misma forma que en la prueba anterior con cuadrados aquí se identificó los falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos para calcular los resultados de las métricas de confusión.

Tabla 4-3: Tabla de la Matriz de Confusión del Detector de Círculos

MATRIZ DE CONFUSIÓN FIGURA CÍRCULO		**Clasificador**	
		Positivos	Negativos
Valor Real	Positivos	23	2
	Negativos	2	13

Realizado por: Castañeda, Ruth V., 2021

Una vez obtenido los datos de la matriz de confusión del círculo se calculó las siguientes métricas:

$$exactitud = \frac{23+13}{23+2+2+13} = 0.9x100 = 90\% \quad (8.2)$$

$$precisión = \frac{23}{23+2} = 0.92x100 = 92\% \quad (9.2)$$

$$sensibilidad = \frac{23}{23+2} = 0.92x100 = 92\% \quad (10.2)$$

$$especificidad = \frac{13}{13+2} = 0.866x100 = 86{,}6\% \qquad (11.2)$$

Con los resultados de los valores de las métricas de la matriz de confusión se determina que el modelo entrenado está siendo sensible antes que específico con un porcentaje en sensibilidad de 92%, con esto evita detectar falsos negativos antes que falsos positivos, no obstante, en cuanto a precisión el modelo tiene un 92% por lo cual se lo considerada bueno para el desarrollo del sistema.

3.2.3 Detector de Triángulos

En la tabla 5-3 se visualiza la matriz de confusión de un triángulo con una muestra de 40 *flashcards*, 25 imágenes correctas y 15 indistintas, de la misma manera que en las pruebas anteriores de cuadrados y círculos, se buscó los falsos positivos, verdaderos positivos, falsos negativos y verdaderos negativos para calcular las métricas de análisis de la matriz.

Tabla 5-3: Tabla de la Matriz de Confusión del Detector de Triángulos

MATRIZ DE CONFUSIÓN FIGURA DE TRIÁNGULOS		**Clasificador**	
		Positivos	Negativos
Valor Real	Positivos	21	4
	Negativos	2	13

Realizado por: Castañeda, Ruth V., 2021

Una vez conseguidos los datos de las figuras triangulares se calculó las métricas de exactitud, precisión, sensibilidad y especificidad.

$$exactitud = \frac{21+13}{21+4+2+13} = 0.85x100 = 85\% \qquad (8.2)$$

$$precisión = \frac{21}{21+2} = 0.91x100 = 91\% \qquad (9.2)$$

$$sensibilidad = \frac{21}{21+4} = 0.84x100 = 84\% \qquad (10.2)$$

$$especificidad = \frac{13}{13+2} = 0.86x100 = 86\% \qquad (11.2)$$

Con los resultados de las métricas de la figura de triángulos se determina que este modelo evita falsos positivos antes que falsos negativos al tener una especificidad de 86% mayor al valor de sensibilidad, con una precisión en aciertos de 91%, por lo cual a este detector se lo considera bueno para el desarrollo del sistema.

Antes de comparar los resultados métricos acertados de los modelos predictivos se visualiza en la figura 4-3 como el Sistema *Happy* ve un falso positivo y un falso negativo, para determinar qué tipo de error le conviene evitar al sistema.

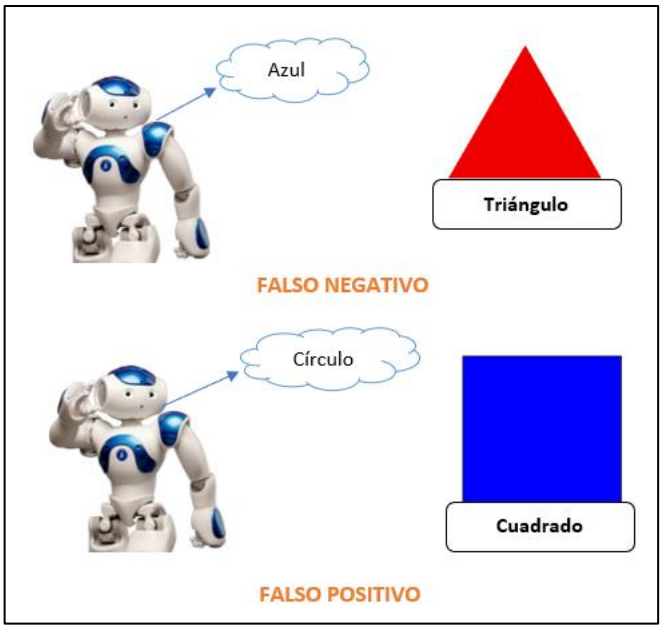
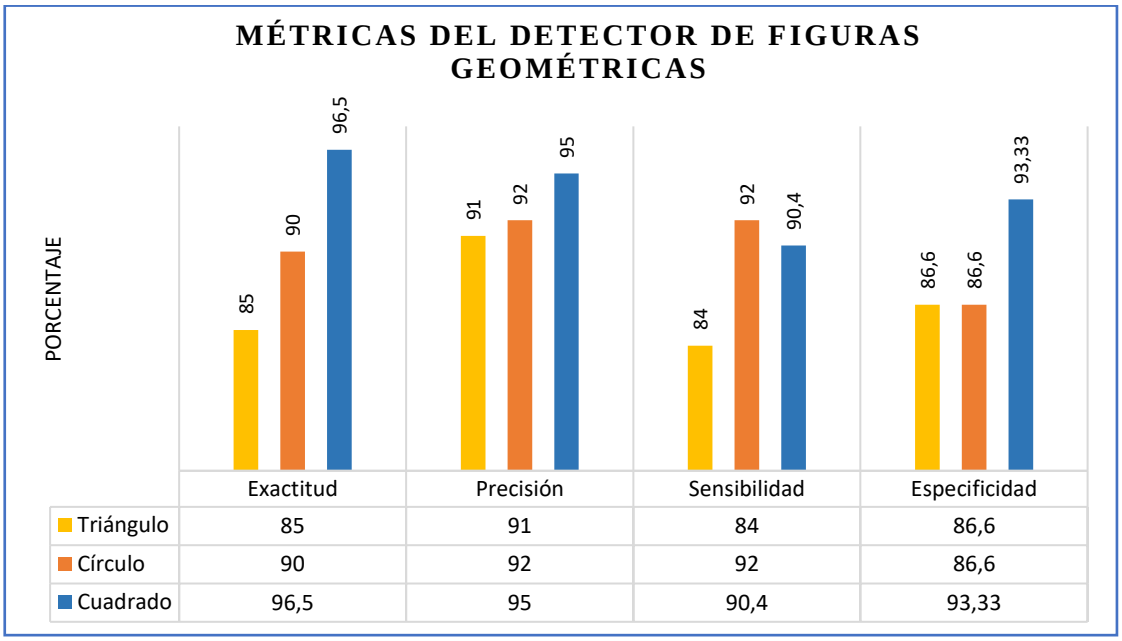


Figura 4-3: Tipos de errores en la Matriz de Confusión

Realizado por: Castañeda, Ruth V., 2021



	Exactitud	Precisión	Sensibilidad	Especificidad
Triángulo	85	91	84	86,6
Círculo	90	92	92	86,6
Cuadrado	96,5	95	90,4	93,33

Gráfico 3-3: Datos métricos del Detector de Figuras Geométricas

Realizado por: Castañeda, Ruth V., 2021

En el gráfico 3-3 se determina que el algoritmo entrenado de triángulos en cuanto a precisión y exactitud es menos previsible en comparación a los demás con un valor de 85% y 91% de aciertos,

no obstante, el algoritmo de círculos tiene un mayor número de falsos positivos con una sensibilidad superior a la especificidad. Con estos datos se determina que los tres módulos predictivos tienen un rango bueno de aciertos, pero el algoritmo de círculos tiene un mayor número de datos equívocos al tener su sensibilidad alta.

3.3 Detector de Figuras con Colores

En esta prueba se unió el detector de colores con el detector de figuras geométricas en un subsistema, de las pruebas anteriores se tomó 52 muestras de *flashcards* para trabajar en un rango de distancia de 40cm a 60cm y verificar el número de aciertos del sistema como se visualiza en la figura 5-3.

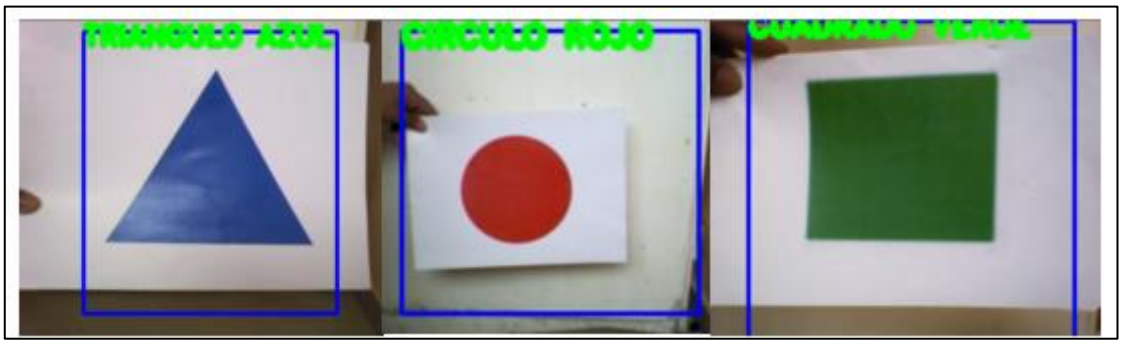


Figura 5-3: Detector de Figuras con colores

Realizado por: Castañeda, Ruth V., 2021

En la tabla 6-3 se determinó el valor porcentual de aciertos del subsistema con un cuadro de doble entrada para las variables fijas colores y figuras geométricas, en la cual se determinó el porcentaje de aciertos con la unión de las dos variables.

Tabla 6-3: Tabla Porcentual del Detector de Figuras con Colores

COLORES	FIGURAS GEOMÉTRICAS		
	TRIÁNGULO	**CÍRCULO**	**CUADRADO**
AMARILLO	82,30%	85,40%	85,80%
VERDE	89,40%	92,10%	92,80%
AZUL	93,70%	95,40%	92,70%
ROJO	92,60%	94,70%	95,70%

Realizado por: Castañeda, Ruth V., 2021

En el gráfico 4-3, se detalla cómo se comportó el subsistema, con estos resultados se determinó que las *flashcards* de color rojo y azul tienen un mayor número de precisión en cualquier figura geométrica usada, de igual manera el algoritmo entrenado de cuadrados y círculos prevalece en exactitud con respecto a los triángulos.

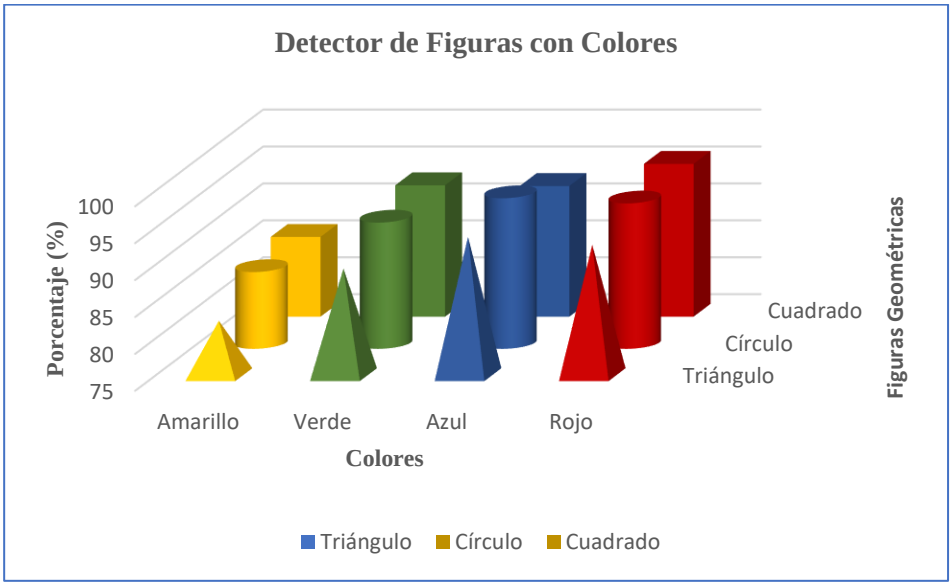


Gráfico 4-3: Detector de Figuras con Colores

Realizado por: Castañeda, Ruth V., 2021

3.4 Prueba de interacción con niños

Para evaluar el sistema *Happy* se realizaron pruebas con niños de 4 años en el Centro de Educación Inicial "San Pablo" en la ciudad de Riobamba con un total de 41 estudiantes, debido a la emergencia sanitaria se tuvo tres clases virtuales y una presencial, donde el robot Nao asistió al docente a través de una metodología interactiva.

3.4.1 Clases Virtuales

El encuentro de los niños con el robot Nao en clases virtuales fue a través de la plataforma *ZOOM* con el uso de la cámara de la computadora en una duración de 20 minutos por clase con la presencia de 41 estudiantes, antes de iniciar el docente explicó la metodología de enseñanza-aprendizaje y la forma que se desarrolló, donde al finalizar a través de un juicio valorativo se les hizo levantar la mano a cada uno para saber si les gusto o no a la clase trabajar con el robot, en la figura 6-3 se evidencia la respuesta de los niños y la manera que estuvo distribuida la actividad junto al docente.

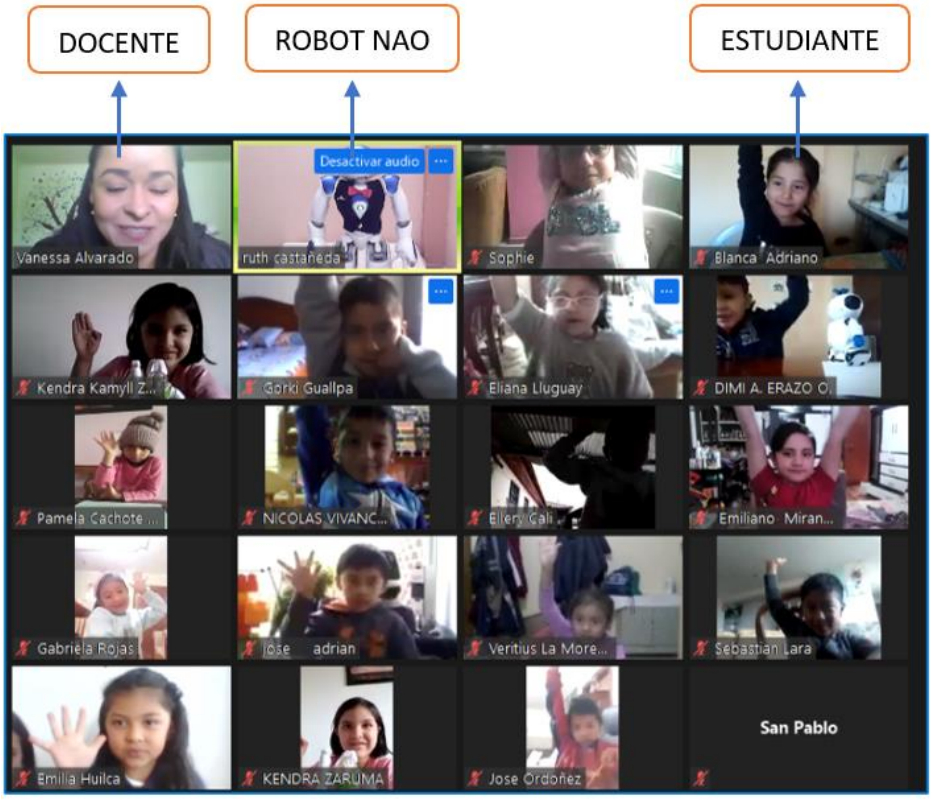


Figura 6-3: Resultado de aceptación por los niños

Realizado por: Castañeda, Ruth V., 2021

Tabla 7-3: Tabla de Aceptación del robot Nao

Juicio Valorativo	Resultados
Les gusto	39
No les gusto	2
Porcentaje de aceptación	95,12%

Realizado por: Castañeda, Ruth V., 2021

En la tabla 7-3 se detalla el resultado de la aceptación del Robot Nao el cual fue de un 95.12%, a la mayoría les gusto la presencia del robot en la actividad de la clase y durante el tiempo de desarrollo estaban atentos y prestos aprender, la participación de los niños fue casi en su totalidad donde se evidenció las ventajas que genera en los estudiantes incluir pedagogías innovadoras.

El sistema *Happy* en el desarrollo de esta clase permitió a los niños reforzar sus conocimientos, se entrenó su memoria visual, se despertó curiosidad por la tecnología al ver un robot humanoide interactuar con ellos, y con respecto al aprendizaje les era fácil para ellos recordar la experiencia a diferencia a una clase con un entorno normal.

3.4.2 Clases presenciales

Las clases presenciales se hicieron con la misma institución educativa, no fue obligatorio la presencia debido a la emergencia sanitaria, sin embargo asistieron 38 estudiantes que se los dividió en tres grupos de trabajo, la didáctica que se manejó fue un juego interactivo con el uso del sistema *Happy* donde los niños tenían que mostrarle al robot Nao las *flashcards* que ellos elijan y verificar si identifica de manera correcta el color o figura, en base al subsistema determinado al inicio como se visualiza en la figura 7-3.

Figura 7-3: Interacción del robot Nao con los niños

Realizado por: Castañeda, Ruth V., 2021

En base al desarrollo de la clase presencial se determinó que la mayoría los niños participaron en la actividad, a estudiantes que se les notaba en un principio aislados con el pasar de la clase se habían integrado al grupo, el robot Nao capto casi toda la atención de los niños. Para valorar como funcionó el sistema *Happy* se hizo preguntas a los niños sobre si les gusto la actividad y si desean tener este tipo de entornos de aprendizaje, ver figura 8-3.

Figura 8-3: Preguntas a los niños sobre el robot Nao

Realizado por: Castañeda, Ruth V., 2021

Con las respuestas de los niños se hace una tabla de interés global, donde se calificó con tres tipos alto, medio y bajo, la respuesta de los niños fue variando, pero todos se sentían captados con este nuevo tipo de enseñanza, con esto se determina que el Sistema *Happy* tiene una buena acogida, como se detalla en la tabla 8-3.

Tabla 8-3: Tabla de Interés Global

Nivel de interés	**Resultados**
Alto	92,30%
Medio	5,30%
Bajo	2,40%
Total	100%

Realizado por: Castañeda, Ruth V., 2021

3.4.3 Valoración del Aprendizaje

En esta prueba se hizo una encuesta a los niños sobre lo que aprendieron con el robot Nao, y según su respuesta se evaluó el desempeño del sistema en el aula de clases, en dos parámetros: Aprendido y No Aprendido.

Tabla 9-3: Tabla Porcentual de Valoración del Aprendizaje

Parámetros	**Sistema *Happy***		
	Colores	**Figuras**	**Figuras con Colores**
Aprendido	95,90%	91,80%	93,70%
No aprendido	4,10%	8,20%	6,30%
Total	100%	100%	100%

Realizado por: Castañeda, Ruth V., 2021

El resultado de la tabla 9-3 en base a la encuesta realizada a los niños, se determina que el subsistema que tuvo mayor acogida fue el de colores, sin embargo, las tres pedagogías se consideran buenas al estar en un rango superior al 91,80%, lo cual determina que el sistema en general está siendo de ayuda en el aprendizaje cognitivo de los niños. En el grafico 5-4 se muestra de forma visual en un esquema de barras el resultado de las preguntas de los niños para identificar el contraste de los tres subsistemas.

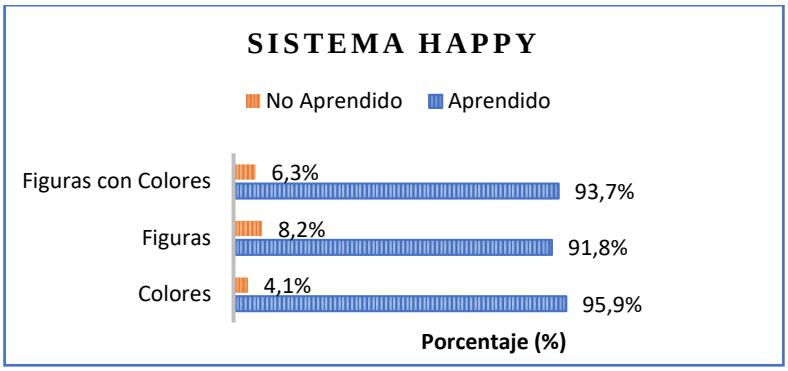


Gráfico 5-3: Resultados de la tabla de Valoración del Aprendizaje

Realizado por: Castañeda, Ruth V., 2021

Con los resultados del gráfico 5-3 se comprueba que el subsistema de figuras tiene una valoración menor con respecto al de colores y figuras con colores, por lo cual se recomienda que al momento de desarrollar las actividades se inicie por el subsistema de colores, debido a que tiene una valoración de aprendizaje del 95,9% y les es más fácil integrarse a los niños con el uso de este subsistema.

3.5 Análisis de Costo del Sistema

El costo del desarrollo del Sistema *Happy* se detalla en la Tabla 10-3, donde se considera que el mayor precio es la adquisición del Robot humanoide sin incluir el IVA. El valor final de la implementación del sistema es de $10 745,30.

Tabla 10-3: Costo del Sistema

Componente del Sistema	Concepto	Inversión ($)
Robot Humanoide Nao (Disponible en la Escuela Superior Politécnica de Chimborazo)	Paquete Robot Humanoide Nao V5 Router, Batería, Licencia de Software	$9 500,00
Ordenador Portátil	Procesador Toshiba Intel Core i7, RAM de 8GB, un Disco duro de 750GB	$ 1 200,00
Flashcards	Tarjetas de Presentación en Formato A4	$ 45,30
Total		$10 745,30

Realizado por: Castañeda, Ruth V., 2021

En base a la tabla 10-3 del análisis de costos del sistema, se determina que el robot Nao en comparación con los mencionados en la tabla 1-1, genera una reducción de costos debido a que al momento de adquirir un robot se debe tomar en cuenta si esta accesible en el país, los costos de importación y las tasas aduaneras, además de verificar si cumple con los requisitos de hardware y software que necesita el sistema.

CONCLUSIONES

- En la implementación del sistema apoyo a la educación parvularia con un robot humanoides utilizando técnicas de visión artificial, se evidenció que es una herramienta pedagógica de ayuda a los docentes, al considerar que logra captar su interés, despertar su curiosidad y darles confianza con el entorno de clase, debido a que ellos consideran la actividad como un juego de aprendizaje, por lo cual se determina que el sistema *Happy* es un método de apoyo en la formación cognitiva de los estudiantes.

- Con las pruebas del detector de colores en entornos controlados y no controlados se establece que el robot Nao con respecto a las *flashcards* debe estar en un rango de distancia entre 40cm a 60cm, para que los módulos de adquisición de datos capturen de manera correcta la información a procesar, y el módulo de voz de una respuesta acertada de colores y figuras.

- En las clases presenciales se concluye que el robot Nao puede trabajar de manera autónoma durante tres clases seguidas de 20 minutos, siempre que se inicie con una carga de 90%, por lo cual se determina que es un robot apto para manejarse con los tiempos de clases de instituciones educativas.

- Las pruebas del detector de figuras geométricas con los datos de la matriz de confusión establecen que los modelos entrenados tienen una precisión de: en triángulos de 91%, en círculos de 92%, y en cuadrados de 95%, y exactitud de: en triángulos de 85%, en círculo 90%, y en cuadrado de 96,5%.

- En clases virtuales a través de una encuesta valorativa se determinó que el 95,12% de los estudiantes les gusta trabajar con el robot Nao, debido a que despierta su curiosidad por explorar conocimientos involucrados con la robótica.

RECOMENDACIONES

- Se recomienda elegir un robot humanoide que tenga una plataforma de desarrollo en código abierto para tener accesibilidad al procesador, dado que hay robots que a pesar de ser comerciales no tienen una interfaz que permite acceder al software.

- Para el uso del sistema *Happy* se recomienda crear grupos de trabajo con máximo 15 estudiantes si es de manera presencial, para que todos logren tener un espacio de interacción con el robot Nao.

- Para el uso de técnicas de visión artificial es preferible usar el software Python, debido a la diversidad de librerías que maneja para el procesamiento de datos y extracción de características.

- El mundo de la robótica cada día está avanzando y se recomienda seguir investigando las ventajas que tiene enseñar con robots en las aulas de clases, para que el sistema educativo tenga nuevas pedagogías de enseñanza-aprendizaje.

BIBLIOGRAFÍA

AGUIRRE DOBERNACK, Nicolás. Implementación de un Sistema de Detección de señales de tráfico mediante Visión Artificial basado en FPGA (Trabajo de titulación) (Grado). Universidad de Sevilla, Departamento de Ingeniería Electrónica, Escuela Superior de Ingenieros. Sevilla-España. 2013. pp. 42-45

ALDEBARAN, *NAO Software 1.14.5 Documentation* [En línea]. 2017. [Consulta: 10 marzo 2021]. Disponible en: http://doc.aldebaran.com/1-14/dev/naoqi/index.html

ALIVEROBOTS, *Los Robots del futuro son ya una realidad* [En línea]. Madrid, España. 2018 [Consulta: 3 de enero 2021]. Disponible en: https://aliverobots.com/nao/

AMIN. *Cascade Triner GUI* [Blog]. 2017. [Consulta: 11 enero 2021]. Disponible en: https://amin-ahmadi.com/cascade-trainer-gui/

BARRERA FLÓREZ, Diana Marcela. Aplicación para reconocimiento de patrones numéricos en contexto educativo, empleando Robot DARWIN-OP [En línea] (Trabajo de titulación). (Grado) Universidad Santo Tomás, Facultad Ingeniería Electrónica. Bogotá-Colombia. 2016. pp 22-24. [Consulta: 23 de febrero 2021]. Disponible en: https://repository.usta.edu.co/bitstream/handle/11634/3487/BarreraDiana2016.pdf?sequence=1&isAllowed=y

BARRERA LOMBANA, Nelson. "Uso de la robótica educativa como estrategia didáctica en el aula de clases". *Red de Revistas Científicas de América Latina, el Caribe, España y Portugal* [en línea], 2015, (Colombia) 6(11), pp. 218-220. [Consulta: 7 febrero 2021]. ISSN 2216-0159. https://www.redalyc.org/articulo.oa?id=477247215010

BARRIOS ARCE, Juan Ignacio. *La matriz de confusión y sus métricas* [Blog]. 2017. [Consulta: 8 marzo 2021]. Disponible en: https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

BOSTONDYNAMICS, The most dynamic humanoid robot [blog]. [Consulta: 14 febrero 2021]. Disponible en: https://www.bostondynamics.com/atlas

BURITICÁ ARBALÁEZ, César Alejandro. SOPHIA, robot humanoide – Hanson Robotics [blog]. 2020. [Consulta: 4 febrero 2021]. Disponible en: https://www.upb.edu.co/es/documentos/doc-boletinescuelaverano2-1464182415182.pdf

CABÁS ORMAECHEA, Luis María. Mecatrónica bioinspirada de robots humanoides de tamaño natural [En linea] (Trabajo de titulación). (Grado) Universidad Carlos III de Madrid, España. 2009. pp 8-11. [Consulta: 10 de marzo 2021]. Disponible en: http://repositori.uji.es/xmlui/bitstream/handle/10234/27826/ICC_2006-10-03.pdf?sequence=1&isAllowed=y

CABRERA BERNAL, Julio César & ZAMBRANO OSTAIZA José Bolívar. Desarrollo de algoritmos de programación para diferentes aplicaciones prácticas en el Robot Bioloid gp [En linea] (Trabajo de titulación). (Grado) Universidad Católica Santiago de Guayaquil, Ecuador. 2016. pp 43-47. [Consulta: 10 de febrero 2021]. Disponible en: http://201.159.223.180/bitstream/3317/6605/1/T-UCSG-PRE-TEC-ITEL-163.pdf

CALVO, Diego. *Clasificación de Redes Neuronales Aritficiales* [blog]. 2017.[Consulta: 20 febrero 2021]. Disponible en: https://www.diegocalvo.es/clasificacion-de-redes-neuronales-artificiales/

CASTELLO, Luis A.; & MÁRSICO, Claudia T. *Diccionario Etimológico de términos usuales en la praxis docente.* Buenos Aires-Argentina: Editorial Altamira, 1995. P. 12

DOXYGEN. *OpenCV, Open Computer Source Vision,* [Blog]. 2018. [Consulta: 11 enero 2021]. Disponible en: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

FERNÁNDEZ MEDINA, Alexander Jorge. Diseño e implementación de un prototipo de clasificación de Lúcumas aplicando Visión Artificial (Trabajo de titulación) (Grado). Universidad Católica de Santa María, Facultad de Ciencias e Ingenierías Física y Formales, Escuela Profesional de Ingeniería Mecánica, Mecánica Eléctrica y Mecatrónica. Arequipa-Perú. 2019. pp. 28-30

GAYO MARTINEZ, Juan Manuel. Diseño de un Brazo Industrial Neumático (Trabajo de titulación) (Grado) Universidad Carlos III de Madrid, Léganes, España. 2013. pp. 15-18

GÓMEZ, Desirée. *El papel de los robots en la educación del futuro* [blog]. [Consulta: 28 noviembre 2020]. Disponible en: https://apel.es/papel-los-robots-la-educacion-del-futuro/

HERVÁS, Carlos; BALLESTEROS, Cristóbal; & CORUJO María C. "La robótica como estrategia didáctica para las aulas de clases de Educación Primaria". *Revista Educativa Hekademos* [en línea], 2018, (España) 24(1), pp. 30-34. [Consulta: 12 enero 2021]. ISSN 1989-3558. Disponible en: https://dialnet.unirioja.es/servlet/articulo?codigo=6542601

IAT20. *Robots Humanoides: Jugando a ser dioses con la robótica* [blog]. [Consulta: 15 enero 2021]. Disponible en: https://iat.es/blog/robots-humanoides/

IBARRA FLORES, William Iván. Sistema de control de acceso mediante identificación y verificación facial fundamentado en algoritmos de aprendizaje automático y redes neuronales [En linea], (Trabajo de titulación). (Grado) Universidad de las Fuerzas Armadas – ESPE, Sangolquí, Ecuador. 2020. pp. 34-36 [Consulta: 15 febrero 2021]. Disponible en: http://repositorio.espe.edu.ec/xmlui/bitstream/handle/21000/22530/T-ESPE-043856.pdf?sequence=1&isAllowed=y

JETBRAINS. *Funcionalidades de PyCharm,* [Blog]. 2021. [Consulta: 23 enero 2021]. Disponible en: https://www.jetbrains.com/es-es/pycharm/features/

LONGONI, María G.; PORCEL, Eduardo A.; LÓPEZ, María V.; & DAPOZO, Gladys N. "Modelos de Redes Neuronales Perceptrón Multicapa Y de Base Radial para la predicción del rendimiento académico de alumnos universitarios". *RedUNCI – CACIC* [en línea], 2010, Argentina, pp.696-697 [Consulta: 12 febrero 2021]. ISSN 978-950-9474-49-9. Disponible en: http://sedici.unlp.edu.ar/handle/10915/19333

LÓPEZ, Pedro A.; & SOSA, Hugo A. *"Aprendizaje de y con robótica, algunas experiencias"* [en línea], 2013, (Colombia) 37(1), p.44. [Consulta: 16 diciembre 2020]. ISSN 2215-2644 Disponible en: https://revistas.ucr.ac.cr/index.php/educacion/article/view/10628/10298

MARÍN, Rafael. *¿Qué es OpenCv? Instalación en Python y ejemplos básicos,* [Blog]. 2020. [Consulta: 28 enero 2021]. Disponible en: https://revistadigital.inesem.es/informatica-y-tics/opencv/

MARTÍNEZ, Marcos. *Robótica para niños: ¿Qué ventajas y beneficios les aporta?* [blog]. [Consulta: 25 noviembre 2020]. Disponible en: https://www.nobbot.com/personas/ventajas-beneficios-robotica-ninos/

MASSIRIS, M.; DELRIEUX, C.; & FERNÁNDEZ A., "Detección de equipos de protección personal mediante red neuronal convolucional YOLO". *En Actas de las XXXIX Jornadas de Automática,* [en línea], 2018, (España), pp. 1023-1025. [Consulta: 2 febrero 2021]. ISSN 978-84-9749-756-5. Disponible en: https://ruc.udc.es/dspace/handle/2183/24891

MINISTERIO DE EDUCACIÓN DEL ECUADOR. *Marco Legal Educativo Ley orgánica de educación intercultural y Reglamento General.* Quito-Ecuador, 2012. pp.15-16

MINISTERIO DE EDUCACIÓN DEL ECUADOR. *Ley orgánica de educación intercultural y Reglamento General.* Quito-Ecuador, 2012. pp.32-35

MONSALVES GONZÁLES, Sara. *"Estudio sobre la utilidad de la robótica educativa desde la perspectiva del docente"* [en línea], 2010, (Chile) 32(90), pp. 90-92. [Consulta: 15 diciembre 2020]. ISSN 0798-9792 Disponible en: https://www.redalyc.org/pdf/659/65920055004.pdf

MORENO A.; ARMENGOL E.; BÉJAR J.; BALANCHE L.; CORTÉS U.; GAVALDÁ R.; GIMENO J.M.; LÓPEZ B.; MARTÍN M; & SÁNCHÉZ M. *Aprendizaje automático* [en línea]. Barcelona-España: Ediciones UPC, 1994. [Consulta: 17 febrero 2021]. Disponible en: https://upcommons.upc.edu/bitstream/handle/2099.3/36157/9788483019962.pdf?sequence=1&isAllowed=y

MORENO, Iveth; MUÑOZ, Lilia; SERRACÍN, José Rolando; QUINTERO, Jacqueline; PITTÍ PATIÑO, Kathia; & QUIEL, Juan. "La Robótica Educativa, una herramienta para la enseñanza – aprendizaje de las ciencias y las tecnologías". *Educación y Cultura de la Información*, vol. 13, n°2 (2012), (España) pp. 76-79.

PEÑAFIEL MERCADO, Iván Darío; MERCADO PÉREZ, Rodrigo Samuel; & PETRO ROMERO, Verónica. Las flash cards como estrategia didáctica en el proceso de aprendizaje de vocabulario de inglés en estudiantes de 2° grado de la Institución Educativa Normal Superior de Sincelejo (Trabajo de titulación) (grado). Corporación Universitaria del Caribe, Facultad de Humanidades y Educación. Sincelejo, Colombia. 2015. pp. 38-40

RAMÍRES BENZAVIDEZ, Kryscia. *Sistemas de Sensación Segmentación, Reconocimiento y Clasificación de Objetos* [Blog]. 2017. [Consulta: 8 enero 2021]. Disponible en: http://www.kramirez.net/Robotica/Material/Presentaciones/VisionArtificial.pdf

REYES, Juan. "Construcción de una Interfaz Eléctrica de Salida para Sistemas de Visión Artificial Basados en MATLAB". *Centro Nacional de Investigación y Desarrollo Tecnológico* [en línea], 2015(México) 1(44), pp.46-50. [Consulta: 20 febrero 2021]. ISSN 968-878-205. Disponible en: https://www.researchgate.net/publication/228974186_Construccion_de_una_Interfaz_Electrica_de_Salida_para_Sistemas_de_Vision_Artificial_Basados_en_MATLAB

ROBLEDANO, Ángel. *Qué es Python: Características,* [Blog]. 2019. [Consulta: 17 febrero 2021]. Disponible en: https://openwebinars.net/blog/que-es-python/

RODRÍGUEZ GARCÍA, Francisco Javier. Diseño y realización de la cabeza de un robot humanoide por impresión 3D a [En línea] (Trabajo de titulación). (Maestría) Escuela Politécnica Superior Ferrol, Coruña, España. 2018. pp. 13-15. [Consulta: 10 enero 2021]. Disponible en: https://ruc.udc.es/dspace/bitstream/handle/2183/22171/RodriguezGarcia_FranciscoJavier_TFM_2018.pdf?sequence=2&isAllowed=y

ROUHIAINEN, Lasse. *Inteligencia artifical 101 cosas que debes saber hoy sobre nuestro futuro.* Barcelona, España: Editorial Planeta, S.A., 2018. 978-84-17568-08-5, pp. 17-24

SANDOVAL, Lilian J. "Algoritmos de Aprendizaje Automático para análisis de predicción de datos". *Revista Tecnológica ITCA-FEPADE* [en línea], 2018, (El Salvador) 11(6), pp. 36-38. [Consulta: 13 febrero 2021]. Disponible en: http://redicces.org.sv/jspui/bitstream/10972/3626/1/Art6_RT2018.pdf

SOFTBANK Robotics, *Pepper y NAO robots para la enseñanza* [En línea]. 2019. [Consulta: 10 febrero 2021]. Disponible en: https://www.softbankrobotics.com/emea/es/pepper-and-nao-robots-education

SOFTBANK Robotics, *NAO* [En línea]. 2018. [Consulta: 14 enero 2021]. Disponible en: https://www.softbankrobotics.com/emea/es/nao

SOFTBANK Robotics, *DOCUMENTATION* [En línea]. 2018. [Consulta: 14 enero 2021]. Disponible en: http://doc.aldebaran.com/2-8/home_nao.html

VIDAL, Marc. *El primer robot que sustituye a un profesor. La urgencia de preparar la relación entre humanos y robots* [blog]. 2016. [Consulta: 28 enero 2021]. Disponible en: https://www.marcvidal.net/blog/el-primer-robot-que-sustituye-a-un-profesor-universitario-la-obligacin-de-preparar-las-relaciones-entre-humanos-y-humanoides

ANEXOS

Anexo A: Conexión del Robot Nao con Python.

```
import cv2
from PIL import Image
import numpy
from naoqi import ALProxy
import  vision_definitions
import imutils
def main():
  IP = "169.254.232.44"
  PORT = 9559
  print "Creating ALVideoDevice proxy to ", IP
  camProxy = ALProxy("ALVideoDevice", IP, PORT)
  camProxy.kCameraSelectID=18
  camProxy.setParam(camProxy.kCameraSelectID, 1)
  #Para cambiar la camara de arriba, o del menton 1 es para la del menton y 0 para la de arriba
  resolution = 2
  resolution = vision_definitions.kQVGA

  colorSpace = 11
  videoclient = camProxy.subscribe("python_GVM1", resolution, colorSpace, 30)

  while(1):
    naoImage = camProxy.getImageRemote(videoclient)
    imageWidth = naoImage[0]
    imageHeight = naoImage[1]
    array = naoImage[6]


    im = Image.frombytes("RGB", (imageWidth, imageHeight), array)
    open_cv_image = numpy.array(im)
    open_cv_image = open_cv_image[:, :, ::-1].copy()

    cv2.imshow('imagen',open_cv_image)
    k = cv2.waitKey(5) & 0xFF
    camProxy.releaseImage(videoclient)
    if k == 27:
        break

  cv2.destroyAllWindows()
  camProxy.unsubcribe(videoclient)

main()
```

Anexo B: Programación en Python de la detección de bordes en figuras geométricas.

```
import cv2
from PIL import Image
import numpy
from naoqi import ALProxy
import vision_definitions

#SE REALIZA LA CONEXIÓN CON EL ROBOT NAO POR MEDIO DE UNA DIRECCIÓN IP
def main():
  IP = "169.254.147.201"
  PORT = 9559
  camProxy = ALProxy("ALVideoDevice", IP, PORT)
  tts = ALProxy("ALTextToSpeech", IP, PORT)
  camProxy.kCameraSelectID=18
  camProxy.setParam(camProxy.kCameraSelectID, 1)
  #Camara del menton 1,y 0 para la de arriba(cabeza)
  resolution = 3 #VGA
  resolution = vision_definitions.kQVGA

  colorSpace = 11 #RGB
  #caprura = cv2-VideoCapture(1)
  videoclient = camProxy.subscribe("python_GVM1", resolution, colorSpace, 30)
  #COLOR ROJO
  redBajo1=numpy.array([175,100,20], numpy.uint8)
  redAlto1=numpy.array([179,255,255], numpy.uint8)
  redBajo2=numpy.array([0,100,20], numpy.uint8)
  redAlto2=numpy.array([3,255,255], numpy.uint8)
  #COLOR AZUL
  azul_bajos = numpy.array([110,100,20])
  azul_altos = numpy.array([130,255,255])
  #COLOR AMARILLO
  amarillo_bajos = numpy.array([21, 100, 20])
  amarillo_altos = numpy.array([35, 255, 255])
  #COLOR VERDE
  verde_bajos = numpy.array([40,100,20])
  verde_altos = numpy.array([80,255,255])

  while(1):
    naoImage = camProxy.getImageRemote(videoclient)
    imageWidth = naoImage[0]
    imageHeight = naoImage[1]
    array = naoImage[6]
#Create A PIL image from our pixel array
    im = Image.frombytes("RGB", (imageWidth, imageHeight), array)
    open_cv_image = numpy.array(im)
    open_cv_image = open_cv_image[:, :, ::-1].copy()
    if True:
        #CONVERTIMOS LA IMAGEN A HSV
        frameHSV = cv2.cvtColor(open_cv_image, cv2.COLOR_BGR2HSV)
        #EXTRAEMOS EL FONDO ROJO Y BUSCAMOS EL CONTORNO
        maskRed1 = cv2.inRange(frameHSV, redBajo1, redAlto1)
        maskRed2 = cv2.inRange(frameHSV, redBajo2, redAlto2)
        maskRed = cv2.add(maskRed1, maskRed2)
        _,contornos,_ = cv2.findContours(maskRed,cv2.RETR_EXTERNAL,
                                         cv2.CHAIN_APPROX_SIMPLE)
        #EXTRAEMOS EL FONDO AZUL Y BUSCAMOS EL CONTORNO
        maskAzul = cv2.inRange(frameHSV, azul_bajos, azul_altos)

        _,contornos1,_ = cv2.findContours(maskAzul, cv2.RETR_EXTERNAL,
                                          cv2.CHAIN_APPROX_SIMPLE)
        #EXTRAEMOS EL FONDO AMARILLO Y BUSCAMOS EL CONTORNO
        maskAmarillo = cv2.inRange(frameHSV, amarillo_bajos, amarillo_altos)
        _,contornos2,_= cv2.findContours(maskAmarillo, cv2.RETR_EXTERNAL,
                                          cv2.CHAIN_APPROX_SIMPLE)
        #EXTRAEMOS EL FONDO VERDE Y BUSCAMOS EL CONTORNO
        maskVerde = cv2.inRange(frameHSV, verde_bajos, verde_altos)
        _,contornos3,_ = cv2.findContours(maskVerde, cv2.RETR_EXTERNAL,
                                          cv2.CHAIN_APPROX_SIMPLE)
```

Anexo C: Programación en Python de los centros y contornos de las figuras a partir de momentos geométricos.

```
        #cv2.drawContours(open_cv_image, contornos, -1, (255,0,0), 3)
        #BUSCAMOS EL CENTRO DE LOS CONTORNOS Y LO CALCULAMOS A PARTIR DE LOS MOMENTOS
        #SE DIBUJA EL CENTRO Y ESCRIBE LAS CORDENADAS DE CADA FIGURA
        for c in contornos:
            area = cv2.contourArea(c)
            if area > 2700:
                M=cv2.moments(c)
                if(M["m00"]==0): M["m00"]==1
                x=int(M["m10"]/M["m00"])
                y=int(M["m01"]/M["m00"])
                cv2.circle(open_cv_image, (x,y),7,(0,255,0),-1)
                font = cv2.FONT_HERSHEY_SIMPLEX
                cv2.putText(open_cv_image,'{},{}'.format(x,y),(x+10,y), font, 0.75,(0,255,0),1,cv2.LINE_AA)
                nuevoContorno = cv2.convexHull(c)
                cv2.drawContours(open_cv_image, [nuevoContorno], 0, (255,0,0), 3)
                tts.say("color rojo")
        for a in contornos1:
            area1 = cv2.contourArea(a)
            if area1 > 2700:
                M1=cv2.moments(a)
                if(M1["m00"]==0): M1["m00"]==1
                x1=int(M1["m10"]/M1["m00"])
                y1=int(M1["m01"]/M1["m00"])
                cv2.circle(open_cv_image, (x1,y1),7,(0,255,0),-1)
                font1 = cv2.FONT_HERSHEY_SIMPLEX
                cv2.putText(open_cv_image,'{},{}'.format(x1,y1),(x1+10,y1), font1, 0.75,(0,255,0),1,cv2.LINE_AA)
                nuevoContorno1 = cv2.convexHull(a)
                cv2.drawContours(open_cv_image, [nuevoContorno1], 0, (255,0,0), 3)
                tts.say("color azul")


        for b in contornos2:
            area2 = cv2.contourArea(b)
            if area2 > 2700:
                M2=cv2.moments(b)
                if(M2["m00"]==0): M2["m00"]==1
                x2=int(M2["m10"]/M2["m00"])
                y2=int(M2["m01"]/M2["m00"])
                cv2.circle(open_cv_image, (x2,y2),7,(0,255,0),-1)
                font2 = cv2.FONT_HERSHEY_SIMPLEX
                cv2.putText(open_cv_image,'{},{}'.format(x2,y2),(x2+10,y2), font2, 0.75,(0,255,0),1,cv2.LINE_AA)
                nuevoContorno2 = cv2.convexHull(b)
                cv2.drawContours(open_cv_image, [nuevoContorno2], 0, (255,0,0), 3)
                tts.say("color amarillo")
        for d in contornos3:
            area3 = cv2.contourArea(d)
            if area3 > 2700:
                M3=cv2.moments(d)
                if(M3["m00"]==0): M3["m00"]==1
                x3=int(M3["m10"]/M3["m00"])
                y3=int(M3["m01"]/M3["m00"])
                cv2.circle(open_cv_image, (x3,y3),7,(0,255,0),-1)
                font3 = cv2.FONT_HERSHEY_SIMPLEX
                cv2.putText(open_cv_image,'{},{}'.format(x3,y3),(x3+10,y3), font3, 0.75,(0,255,0),1,cv2.LINE_AA)
                nuevoContorno3 = cv2.convexHull(d)
                cv2.drawContours(open_cv_image, [nuevoContorno3], 0, (255,0,0), 3)
                tts.say("color verde")
        #cv2.imshow('imagen',maskRed)
        #SE MUESTRA LA IMAGEN Y MENCIONA EL COLOR DE LA FIGURA
        cv2.imshow('im', open_cv_image)
        k = cv2.waitKey(5) & 0xFF
        camProxy.releaseImage(videoclient)
        if k == 27:
            break
    cv2.destroyAllWindows()
    camProxy.unsubcribe(videoclient)

main()
```

Anexo D: Algoritmo de aprendizaje con las extensiones *XML*, en el sistema *Happy*.

```
  #Modulos del Algoritmo entrenado
  clasificar = cv2.CascadeClassifier('cascade.xml')
  clasificar1 = cv2.CascadeClassifier('cascade1.xml')
  clasificar2 = cv2.CascadeClassifier('cascade2.xml')

  while(1):

    naoImage = camProxy.getImageRemote(videoclient)
    imageWidth = naoImage[0]
    imageHeight = naoImage[1]
    array = naoImage[6]

    im = Image.frombytes("RGB", (imageWidth, imageHeight), array)
    open_cv_image = numpy.array(im)
    open_cv_image = open_cv_image[:, :, ::-1].copy()
    gray = cv2.cvtColor(open_cv_image, cv2.COLOR_BGR2GRAY)

    toy = clasificar.detectMultiScale(gray,
                    scaleFactor=5,# 11 cámara compu y 5 camara NAO
                    minNeighbors=100,
                    minSize=(70, 70)  # tamaños de objetos a detectar
                                      )
    for (x, y, w, h) in toy:

        cv2.rectangle(open_cv_image, (x, y), (x + w, y + h), (255, 0, 0), 2)
        cv2.putText(open_cv_image, 'CUADRADO', (x, y + 10), 2, 0.7, (0, 255, 0), 3, cv2.LINE_AA)
        tts.say("ES UN CUADRADO")

    toy1 = clasificar1.detectMultiScale(gray,
                    scaleFactor=5,  # 11 cámara compu y 5 camara NAO
                    minNeighbors=80,
                    minSize=(60, 80)  # tamaños de objetos a detectar
                                      )
    for (x, y, w, h) in toy1:
        cv2.rectangle(open_cv_image, (x, y), (x + w, y + h), (255, 0, 0), 2)
        cv2.putText(open_cv_image, 'TRIANGULO', (x, y + 10), 2, 0.7, (0, 255, 0), 3, cv2.LINE_AA)
        tts.say("ES UN TRIANGULO")


    toy2 = clasificar2.detectMultiScale(gray,
                    scaleFactor=6,# 11 cámara compu y 5 camara NAO
                    minNeighbors=8,
                    minSize=(70, 70)  # tamaños de objetos a detectar
                                      )
    for (x, y, w, h) in toy2:

        cv2.rectangle(open_cv_image, (x, y), (x + w, y + h), (255, 0, 0), 2)
        cv2.putText(open_cv_image, 'CIRCULO', (x, y + 10), 2, 0.7, (0, 255, 0), 3, cv2.LINE_AA)
        tts.say("ES UN CIRCULO")

    cv2.imshow('imagen',open_cv_image)
    k = cv2.waitKey(5) & 0xFF
    camProxy.releaseImage(videoclient)
    if k == 27:
        break
  cv2.destroyAllWindows()
  camProxy.unsubcribe(videoclient)

main()
```

Anexo E: Captura de las imágenes positivas y negativas para entrenarlas

```
Datos = 'n'
if not os.path.exists(Datos):
    print ('Carpeta creada: ', Datos)
    os.makedirs(Datos)

cap = cv2.VideoCapture(0)

X1,Y1 = 190,80
X2,Y2 = 450, 398

count = 0

while True:
    ret,frame = cap.read()
    if ret == False:
        break
    imAux = frame.copy()
    cv2.rectangle(frame,(X1,Y1),(X2,Y2),(255,0,0),2)

    object = imAux[Y1:Y2,X1:X2]
    object = imutils.resize(object, width=38)

    k = cv2.waitKey(1)
    if k == 27:
        break

    elif k == ord('s'):
        cv2.imwrite(Datos+'/object_{}.jpg'.format(count),object)
        print ('Imagen almacenada: ', 'object_{}.jpg'.format(count))
        count = count + 1

    cv2.imshow('frame', frame)
    #cv2.imshow('frame', object)

    k= cv2.waitKey(1)
    if k==27:
        break

cap.release()
```

Anexo F: Entrenamiento en cascada con el programa *CASCADE-TRAINER-GUI*

```
**************************************************
*************** CREATING SAMPLES *****************
**************************************************
Object : C:/Users/Ruth Viviana/PycharmProjects/CrearImagenesPNCIRCULO
Fixing file names in negative images folder.
Fixing file names in positive images folder.
Creating negative list C:/Users/Ruth Viviana/PycharmProjects/CrearImagenesPNCIRCULO/neg.lst
Creating positive list C:/Users/Ruth Viviana/PycharmProjects/CrearImagenesPNCIRCULO/pos.lst
Running : opencv_createsamples
Info file name: C:\Users\Ruth Viviana\PycharmProjects\CrearImagenesPNCIRCULO\pos.lst
Img file name: (NULL)
Vec file name: C:\Users\Ruth Viviana\PycharmProjects\CrearImagenesPNCIRCULO\pos_samples.vec
BG  file name: (NULL)
Num: 300
BG color: 0
BG threshold: 80
Invert: FALSE
Max intensity deviation: 40
Max x angle: 1.1
Max y angle: 1.1
Max z angle: 0.5
Show samples: FALSE
Width: 38
Height: 36
Max Scale: -1
Create training samples from images collection...
Done. Created 300 samples


**************************************************
************* TRAINING CLASSIFIER ****************
**************************************************
Running : opencv_traincascade
PARAMETERS:
cascadeDirName: C:\Users\Ruth Viviana\PycharmProjects\CrearImagenesPNCIRCULO\classifier
vecFileName: C:\Users\Ruth Viviana\PycharmProjects\CrearImagenesPNCIRCULO\pos_samples.vec
bgFileName: C:\Users\Ruth Viviana\PycharmProjects\CrearImagenesPNCIRCULO\neg.lst
numPos: 300
numNeg: 500
numStages: 20
precalcValBufSize[Mb] : 1024
precalcIdxBufSize[Mb] : 1024

acceptanceRatioBreakValue : -1
stageType: BOOST
featureType: HAAR
sampleWidth: 38
sampleHeight: 36

boostType: GAB
minHitRate: 0.995
maxFalseAlarmRate: 0.5
weightTrimRate: 0.95
maxDepth: 1
maxWeakCount: 100
mode: BASIC
Number of unique features given windowSize [38,36] : 908928

===== TRAINING 0-stage =====
<BEGIN
```