Diseño y construcción de un robot cartesiano de 3 grados de libertad

Julie Berrio, MSc [1], Edgar Arcos, MSc [2],
Juan Zuluaga [3], Sergio Corredor [4]

(1) julie.berrio@uac.edu.co
(2) edgar.arcos@uac.edu.co
(3) juan.zuluaga54@uautonoma.edu.co
(4) sergio.corredor@uautonoma.edu.co

Grupo de Investigación en Ingeniería Mecatrónica
Programa de Ingeniería Mecatrónica
Universidad Autónoma del Caribe
Barranquilla, Colombia

Diseño y construcción de un robot cartesiano de 3 grados de libertad

Palabras clave:
Control robótico; procesamiento de señales acústicas; reconocimiento de voz; diseño de software; metodología de diseño; redes neuronales

RESUMEN

Las asignaturas de diseño mecatrónico se centran en el diseño funcional de sistemas autónomos e inteligentes, aplicando diferentes disciplinas tecnológicas (dinámica mecánica, la electrónica de baja y alta potencia, electromecánica, control de movimiento, la óptica, metrología y procesamiento de señales) en un entorno bien equilibrado fortaleciéndose con la realización de prototipos. El presente artículo expone el proceso de diseño, construcción y puesta en marcha de un robot cartesiano con 3 grados de libertad y la programación de su respectivo software de interfaz con el usuario, este fue realizado en lenguaje Python donde las principales ventajas para su escogencia fueron su versatilidad, portabilidad y documentación, además de ser Open-Source. Se requería implementar un algoritmo para el reconocimiento de voz de tal forma que pudiera clasificar 3 palabras diferentes a través de redes neuronales artificiales.

I. INTRODUCCIÓN

Hoy en día, las grandes industrias para ser competitivas en el mercado actual (cambiante y flexible), se han visto favorecidas con el aumento de su productividad al organizar de forma autónoma sus procesos de producción en lo que es llamado "Celdas Flexibles de Manufactura" [1].

El presente proyecto va dirigido al diseño y construcción de un robot cartesiano con fines educativos para la Universidad Autónoma del Caribe, donde se evidencie la implementación del diseño de sistemas mecatrónicos (mecánica, electrónica y programación), procesamiento de imágenes para la replicación de figuras, el transporte de objetos dentro del área de trabajo de este (en nuestro caso, cajas de fósforos), y por último la implementación de sistemas inteligentes, que usando redes neuronales se provea al autómata de la capacidad de reconocimiento de palabras en un archivo de audio, implementando una rutina prevista en el código para cada palabra.

El autómata consta de 3 ejes, designados para dibujar figuras geométricas predeterminadas, replicar dibujos como también realizarlos de forma libre por el usuario. La alimentación se realiza a través de una fuente de voltaje de computador, la cual permite obtener los voltajes de 12, -12, 5 y -5 voltios, esta se encuentra sustentada por corriente alterna de 120 V a 60 Hz. El dispositivo incluye un sistema de tracción para los motores que trabajan a 12V y 6V, donde el robot es capaz de alcanzar velocidades de 0.42 cm/s mientras realizan movimientos precisos [2], para así obtener dibujos consistentes. Posee cuatro motores paso a paso, que proveen el movimiento en tres ejes, y todo esto reposa sobre una base de madera, además cuenta con un Arduino UNO como tarjeta transmisora de datos y requiere de un computador (no incluido) para la interfaz con el usuario. Las dimensiones del robot son 50 cm de ancho, 50 cm de largo y 30 cm de altura con un peso de 10 Kg.

La programación se realizó en el lenguaje de programación Python, el software toma los datos ingresados en la interfaz de usuario, ya sea de forma manual o cargando un dibujo en archivo .PNG. La comunicación entre Python y el robot cartesiano se realizó mediante el Uso de Arduino por el puerto serial.

Este documento inicialmente expondrá el marco teórico referente a el lenguaje de programación implementado, redes neuronales artificiales y el inicio de este tipo de sistemas mecatrónicos, seguido de la metodología utilizada para el diseño, construcción y puesta en marcha del robot cartesiano, se presentan los resultados obtenidos en las pruebas de dibujado como de reconocimiento de palabras usando redes neuronales artificiales y por último las conclusiones son presentados en último apartado.

II. REFERENTE TEÓRICO

En casi todas las épocas y culturas, los hombres han intentado construir máquinas automáticas que faciliten su trabajo, satisfagan su curiosidad y afán de aprender e investigar. En la antigua Grecia ya se habían construido ingeniosos autómatas; luego en la Edad Media y en el renacimiento se siguieron fabricando diversos autómatas, como el gallo de Estrasburgo en 1352, que es el más antiguo autómata que se conserva en la actualidad [3]. A lo largo de los años se han denominado varias posibles configuraciones para un robot, las cuales son: robot cilíndrico, robot esférico o polar, robot angular, robot SCARA, robot paralelo, y en nuestro caso, robot cartesiano.

Los robots cartesianos presentan una estructura articulada, es decir se encuentra conformada por una serie de elementos o eslabones que facilitan el movimiento, el cual puede ser de translación vertical, horizontal y transversal, y pueden combinar movimiento de giro sobre su mismo eje [3]. Por lo general este tipo de autómatas poseen 3 grados de libertad (GDL) los cuales son representados por cada eje x, y, z.

Los robots cartesianos presentan ventajas frente a las demás configuraciones mencionadas (cilíndrica, esférica, angular, Scara, Paralelo), como la precisión uniforme en todo su espacio de trabajo y alta fiabilidad para seguir trayectorias previamente especificadas.

Uno de los requerimientos fue el reconocimiento de 3 palabras usando redes neuronales artificiales, para luego realizar una figura geométrica de las 3 posibles, esta ilustración se encontraba definida por un algoritmo que enviaba movimientos por el puerto serial hacia el Arduino, por último este enviaba los pasos para cada motor paso a paso.

El reconocimiento de palabras por computadora es una tarea compleja de reconocimientos de patrones y de los sistemas biométricos [4]. Por lo regular, la señal de voz se muestra en un rango de 8 a 16 KHz. Para este caso se tomó una frecuencia de muestreo de 22050 Hz grabando 2 segundos a un canal.

A manera de resumen, dentro de la tarea de extracción de características para el reconocimiento de palabras se encuentran los siguientes métodos:

- Análisis de Fourier [5].
- Codificación Predictiva Lineal [6].
- Análisis de coeficientes Cepstrales [7].
- Predicción Lineal Perceptiva [8].

Para el presente trabajo se decidió escoger el Análisis de Fourier, por su facilidad de implementación en el lenguaje de programación Python y presentar mayor documentación y mejor desempeño en diferentes estudios [4].

A. Transformada Discreta de Fourier

Desde la publicación de Cooley y Turkey en 1965 [9] sobre la creación de un algoritmo para el cálculo de la Transformada Discreta de Fourier (DFT) de una forma más eficiente en el desarrollo de procesamiento digital de señales [9], se ha visto el incremento de algoritmos que trabajen la Transformada Rápida de Fourier (FFT) con mayor versatilidad, siendo esta transformada un algoritmo para computar la DFT y su inversa. En el análisis de Fourier se convierte el tiempo (o espacio) a un rango frecuencial o viceversa, en la siguiente imagen se muestra la fórmula de una DFT.

Ecuación 1. Transformada Discreta de Fourier

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \qquad k = 0,\ldots, N-1 \qquad (1)$$

Luego de realizar el Análisis de Fourier, se deben escoger los picos de la señal, y tomar la frecuencia respectiva en la que se encuentran estos, así obteniendo lo que se le conoce en el reconocimiento de palabras como los formantes, para ingresarlos en una red neuronal artificial previamente entrenada.

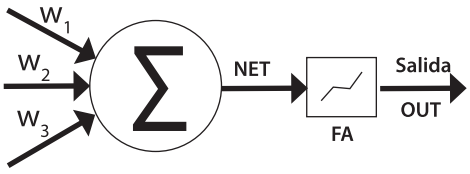
B. Red Neuronal Artificial

Sistema de interconexión de neuronas artificiales que colaboran entre sí para producir un estímulo de salida, por lo general se compone de unidades llamadas neuronas y cada neurona recibe una serie de entradas a través de interconexiones y emite una salida que viene dada por una función de propagación, una de activación y una de transferencia. La neurona artificial se encuentra inspirada en las neuronas biológicas.

Para el presente proyecto, se procedió a crear una Red Neuronal Artificial multicapa, retropropagada, la capa de entrada estaba compuesta por 5 neuronas (5 valores picos del Análisis Espectral de Fourier), la capa oculta número 1 poseía 50 neuronas, la siguiente 6 neuronas y la capa de salida 2 neuronas, el DataSet, se encontraba constituido por 20 muestras para cada palabra lo que sumaba 60 patrones de entrenamiento para toda la Red Neuronal Artificial.

McCulloch y Pitts son considerados como los padres de las Redes Neuronales Artificiales debido a que fueron los primeros en diseñar una neurona artificial, ellos proponen que la neurona es un dispositivo binario con un umbral fijo que se debe superar para un cambio de estado. Recibe sinapsis que excitan otros elementos y tienen una característica de ser de un valor similar [10].

Fig.1. Neurona Artificial [11]



Dentro de los requerimientos del proyecto, se encontraba el escoger un lenguaje de programación

Open-Source, para este caso, se presentaron varios como Java, Python, C, C++, C# y Javascript. En este caso se seleccionó Python por versatilidad, portabilidad, documentación y fácil conexión con otros programas como Arduino y Matlab.

C. Python

Python es un lenguaje de programación poderoso y de fácil uso. Cuenta con estructuras de datos eficientes y de alto nivel, además de un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su "tipado" dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas [12].

III. METODOLOGÍA

Para el diseño, construcción y puesta en marcha del robot cartesiano, usando Python como lenguaje de programación, construyendo y entrando una red neuronal artificial para el reconocimiento de tres palabras, se optó por separar el proyecto en diferentes secciones debido a su alta complejidad, como lo fue, interfaz gráfica, transmisión serial, procesamiento de imágenes, procesamiento de voz y diseño y construcción física. El sistema tiene 4 formas de trabajar las cuales son:

1. Toma los datos ingresados por el usuario desde la interfaz, ya sea para realizar un dibujo predefinido (cuadro, triángulo y círculo) o un esquema deseado en archivo .PNG, luego el robot cartesiano procede a dibujar.
2. Se activa el control manual del robot cartesiano, proveyendo al usuario del control total en los 3 ejes del autómata.
3. Se activa el control automático para carga y descarga de cajas, de esta forma con una rutina predefinida, el robot cartesiano realiza un conjunto de movimientos dentro de su área de trabajo.
4. Se activa el reconocimiento de palabras. En un lapso de dos segundos el usuario debe decir una de las tres palabras (cuadro, triángulo o círculo), luego se presenta un cuadro de texto de con la opciones "Palabra Correcta" o "Palabra incorrecta y así se verifica de forma manual que dicha palabra sea correcta, por último el robot cartesiano realiza la figura.

A. Transmisión serial

Los datos entre el sistema físico y virtual deben ser transmitidos usando una tarjeta de envío de datos, en este caso se usó un Arduino Uno, encargado de enviar las señales a los controladores de los motores paso a paso, luego enviar los "pasos" respectivos a cada motor, ya sea el del eje x (2 motores), el motor del eje y o del eje z.

El algoritmo en Python se encuentra encargado de realizar el procesamiento de voz e imágenes y ejecución manual del autómata. Debe haber una conexión entre Python y el robot cartesiano, pero debido a la facilidad que hubo entre la manipulación del robot desde Arduino, se optó por enviar los datos primeramente hacia esta tarjeta, por el puerto serial, tomarlos, procesarlos y remitir una salida para los motores paso a paso. En la mayoría de los casos se usó el puerto serial 'COM4' a una velocidad de 9600 baudios.

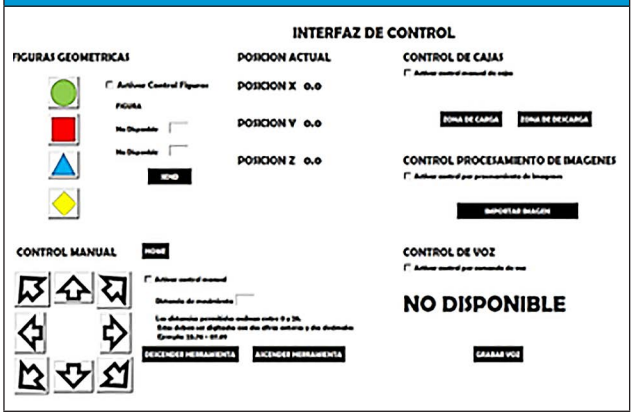
B. Interfaz Gráfica

La interfaz se realizó en Python usando las librerías: Cv2, Tkinter, tkFileDialog, time, re, image e imageTk, a esta interfaz se le agregaron avisos de alertas y errores, por si en algún caso el usuario agregaba datos erróneos, un ejemplo de esto es el escribir letras donde se debe escribir números. La ventana posee una resolución de 1300x700.

La interfaz gráfica se encuentra dividida en diferentes bloques, como: figuras geométricas que son generadas automáticamente (programado dentro del algoritmo general), control manual para los 3 ejes del robot cartesiano, casilla para activar el control manual, sección de monitoreo de posición actual del autómata en su área de trabajo, control automático de cajas (requerimiento de la construcción), donde se recojan en un punto definido por el usuario y se depositen en otro punto, control de procesamiento de imágenes y por último control de

voz para activar el dibujado de figuras geométricas de forma automática.

Fig. 2. Interfaz gráfica



C. Procesamiento de imágenes

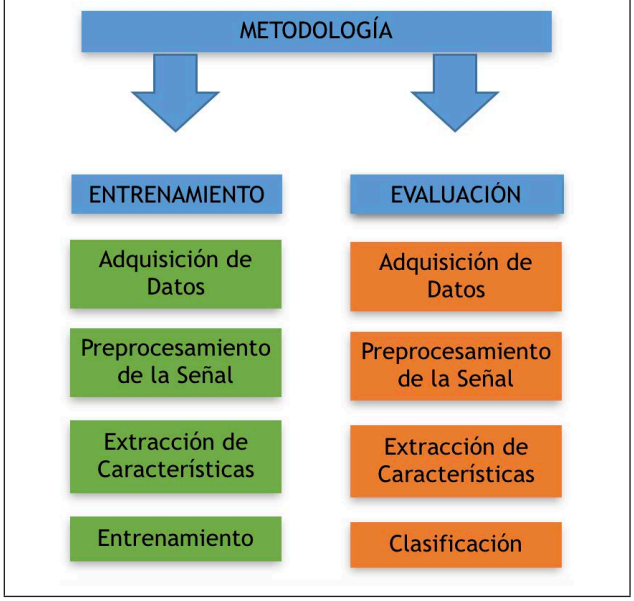
La librería cv2 fue la seleccionada, encargada de todo el procesamiento de imágenes, es similar al Toolbox "*Image Processing Toolbox*" de Matlab presentando funciones como "imred", "imshow", entre otros. Primero se lee la imagen deseada, esta debe tener exactamente un tamaño de 280 x 280 pixeles, la figura a tomar debe ser dibujada en *Paint* con la línea más delgada y posteriormente grabada con extensión .PNG. Los dibujos son tomados y pasados a una matriz de 280 filas x 280 columnas, de la cual va estar llena de 0's en todas las casillas donde se encuentre un punto blanco y 1's donde se encuentre punto negro, luego se envían por el puerto serial las distancias que debe ejecutar el autómata en los ejes Y y X. El puerto serial toma estas distancias y las convierte en número de pasos, los cuales van a ser enviados a los controladores de los motores paso a paso, y así realizar el dibujo requerido.

D. Procesamiento de voz

Se requiere que el robot cartesiano realice tres figuras geométricas (cuadro, círculo y triángulo) que con solo decir la palabra frente al micrófono del computador se reconozco la palabra por medio de un sistema experto, se realizó un enlace directamente con el sector de "figuras geométricas" para que luego de reconocer la palabra se provea desde el teclado las dimensiones de nuestra ilustración.

Las librerías implementadas fueron Pybrain [13] para construir el DataSet, el cual constaba de 20 muestras para cada palabra lo que sumaba 60 patrones de entrenamiento para toda la red, la librería Neurolab encargada de crear la red neuronal artificial, donde se ingresa el DataSet para entrenarla; a lo largo de todo el código se usó SciPy y Numpy por su versatilidad en el manejo de vectores y matrices, como también la función "scipy.io.wavfile" para captar la voz de las personas, por último se empleó el Análisis Espectral de Fourier con la librería Obspy [14,15], la cual contiene funciones como "fft", "dft", "nextpow2", entre otras.

Fig. 3. Metodología para el entrenamiento y clasificación de la palabra [16]



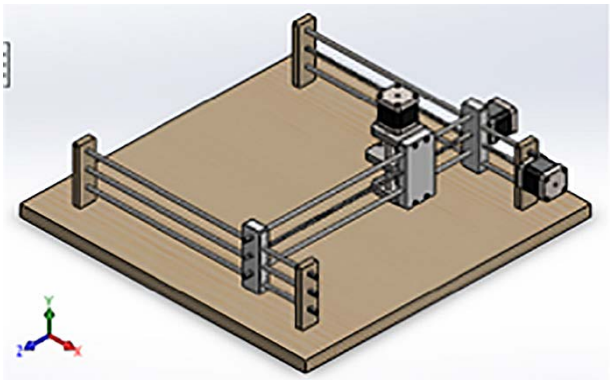
Para esta sección del proyecto se procedió a usar una red neuronal artificial multicapa, retropropagada, la capa de entrada estaba compuesta por 5 neuronas (5 valores picos del Análisis Espectral de Fourier), la capa oculta número 1 poseía 50 neuronas, la siguiente 6 neuronas y la capa de salida 2 neuronas correspondiente a una combinación [0,0] Cuadro, [0,1] Círculo y [1,0] Triángulo, en la figura 3 se puede observar la metodología propuesta.

E. Diseño

Para el diseño del autómata se usó *Solidworks* para tener una visión previa de las dimensiones del

robot cartesiano, sus puntos críticos, puntos límites del área de trabajo, peso aproximado con los materiales propuestos y acople de los motores dentro del esqueleto, en la figura 4 se puede observar el diseño en *Solidworks*, empleando en el eje Y y X, dos varillas de aluminio en el extremo superior e inferior, y una tornillo sin fin en el centro, encargado del movimiento, todo esto actuando como una guía.

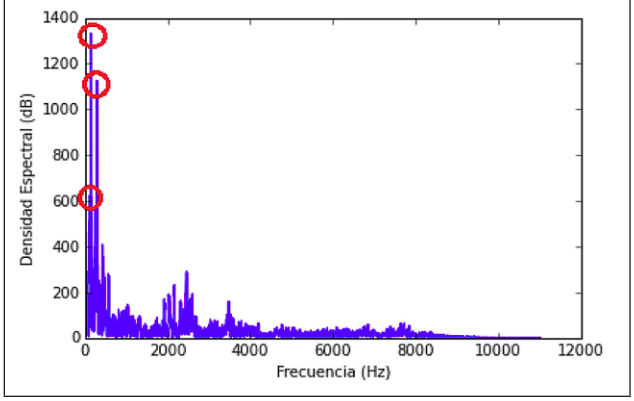
Fig. 4. Diseño y Simulación en Solidworks del Robot Cartesiano



IV. RESULTADOS

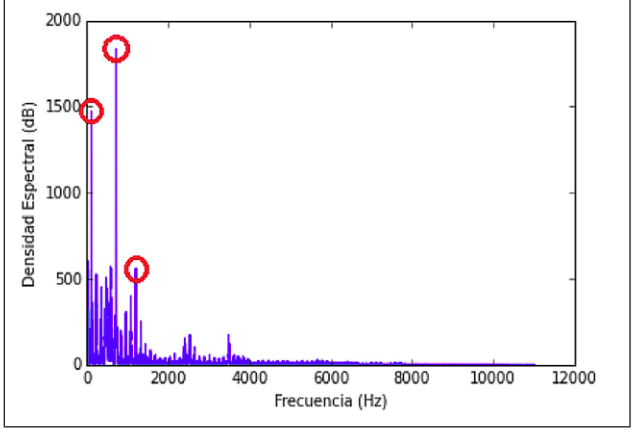
Se logró determinar en el proceso que el error aproximado en el cual el autómata incurría al momento de dibujar un triángulo equilátero, fue de 0.01 centímetros, este por la utilización de motores paso a paso (200 pasos por vuelta) lo que equivale a un ángulo de paso de 1.8°, además por la arquitectura de construcción general del robot cartesiano. El software adquiere datos del usuario para realizar los dibujos correspondientes a las imágenes referidas, debido a esto se hizo necesario analizar la figura, convertirla a escala de grises, ingresar la información en una matriz de 280 x 280 e indicarle a el robot por medio del puerto serial los pasos que debía realizar cada motor paso a paso. La sección del control por voz, debía identificar una palabra entre 3 posibles, como lo era, cuadro, círculo y triángulo. El análisis para diferenciar cada palabra se realizó desarrollando un algoritmo que adquiere el sonido desde el micrófono del computador, en una muestra a 22050 Hz por 2 segundos y 1 canal, que a su vez genera de 44100 muestras, luego se le aplica el Análisis Espectral de Fourier, en este se observó que en una frecuencia cercana a 0 Hz se presentaba una densidad espectral muy alta, la cual perjudicaba la señal, este error se considera el valor DC, por lo cual se le restó a la misma el "*mean value*", por último se realiza la Transformada Rápida de Fourier (fft) usando la librería Obspy, se obtiene una señal representada en: eje de las ordenadas y eje de las abscisas referido a la densidad espectral y la frecuencia, respectivamente.

Fig. 5. Gráfica de la densidad espectral vs frecuencia, palabra "círculo"



En la Figura 5 se puede observar los 3 puntos claves o picos de la señal muestreada y procesada para la palabra círculo, en el eje y, tenemos la densidad espectral, en el eje x se observa que las frecuencias se desempeñan entre 100 y 4000 Hz.

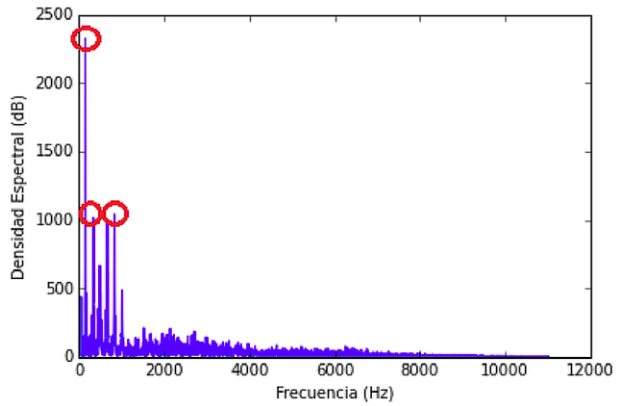
Fig. 6. Gráfica de la densidad espectral vs frecuencia, palabra "cuadro"



En la figura 6 se encuentran los 3 picos fundamentales de la palabra "cuadro", el algoritmo

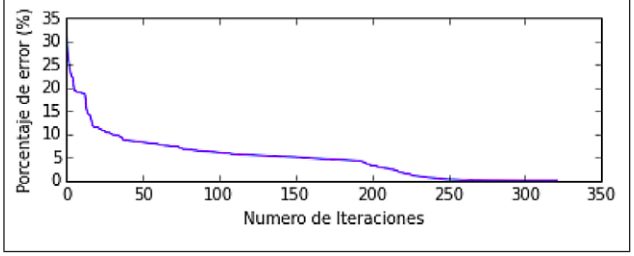
diseñado en Python toma automáticamente los 5 picos de la señal muestreada, con estos 5 valores se crea un patrón de entrenamiento o señal a clasificar en nuestra red neuronal artificial (validación).

Fig. 7. Gráfica de la densidad espectral vs frecuencia, palabra "triángulo"



De esta misma forma se hace para cada una de las 20 palabras y cada una de las opciones (círculo, cuadro y triángulo), con 60 patrones de entrenamiento se construyó un Dataset para ser ingresado en la red neuronal retropropagada y por último solo queda realizar el test. El algoritmo para la red neuronal artificial fue diseñado con un máximo de 1000, un "goal" (error máximo) de 0.02, porque el esperar hasta que convergiese demandaba mucho más tiempo, con una meta de "0.02" el entrenamiento demoró alrededor de 15 segundos. En la figura 8 se muestra el error inicial, el número de épocas que tomó llegar a este error.

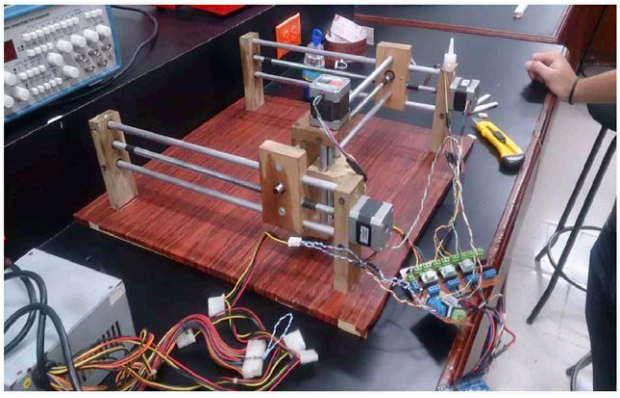
Fig. 8. Error vs Iteraciones



El robot cartesiano realizó todas las figuras geométricas precargadas y todas las figuras que se le cargaban desde un dibujo de Paint, en la siguiente imagen se muestra la estructura física que consta de:

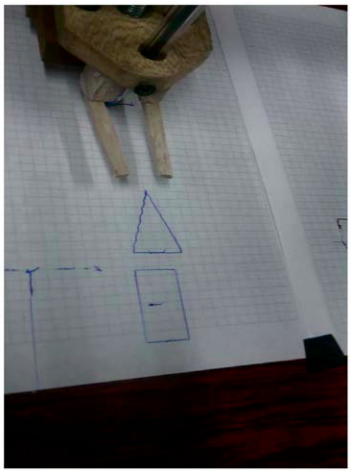
- Base de madera de 50 cm de ancho por 50 cm de largo.
- 4 motores paso a paso referencia NEMA 17, encargados del movimiento, 2 para el eje x, 1 para el eje y por último uno para el eje z.
- 4 controladores Pololu A4988 para los motores paso a paso.
- Arduino UNO

Fig. 9. Estructura física y electrónica



La estructura física y electrónica se puede detallar de mejor manera en la figura 9, donde se pueden ver los motores paso a paso, estructura física hecha en madera y placa controladora de motores.

Fig.10. Realización de figuras predeterminadas



La figura 10 hace referencia a dibujos realizados por el robot cartesiano, el triángulo fue definido desde la interfaz de usuario con una base de 3 cm

y una altura de 4 cm. El rectángulo fue definido con una base de 2.5 cm y una altura de 6 cm. Cuando el robot va a realizar un dibujo, este se ubica en el centro de la figura, convirtiendo este en el punto (0,0) y se dirige luego al punto más cercano a dibujar, ya sea a la derecha, izquierda, arriba o abajo.

V. CONCLUSIONES

El desarrollo de un robot cartesiano conjugó una gran cantidad de ramas de la ingeniería como lo fue, la mecánica, electrónica, programación, sistemas inteligentes, diseño mecatrónico, control, entre otros, por lo cual este es considerado un proyecto integrador de gran envergadura. La utilización de software CAD como paso previo a la construcción de sistemas mecánicos, electrónicos y/o mecatrónicos es de gran beneficio, ya que es lo más cercano a la misma construcción física.

Se concluye que la utilización de motores paso a paso, disminuyó la generación de error en el trazado final de todas las ilustraciones, ya que es ampliamente conocido que estos motores poseen una exactitud mayor comparados con otras configuraciones de motores, los demás materiales empleados en el robot cartesiano son frecuentes dentro de la industria pertinente.

Se puede concluir que el desarrollo de sistemas mecatrónicos usando Python como lenguaje de programación trae grandes ventajas, como la eficacia del proceso, velocidad de computo, excelente y fácil manejo de matrices, fácil acople con otros programas como Arduino, aparte de ser *open-source*.

Se concluye que la red neuronal artificial retropropagada es muy versátil para sistemas de reconocimiento de voz y palabras, ya que permite alcanzar porcentajes de acierto de hasta 99% [4].

REFERENCIAS

[1] Nampring, N. Punglae, S. The future of industrial automation flexible manufacturing systems (FMS). TIM 690 Seminar in Technology and Innovation Management. Graduate school of Management and Innovation King Mongkut's University of Technology Thonburi. 2005

[2] Codina A. Posicionamiento y proyección actual del motor de paso en aplicaciones industriales. Agrupación de Instrumentación, Centro de Inmuno-ensayo. Habana, Cuba. 2012.

[3] Segovia, J. Alamilla, M. Domínguez, J. Robot cartesiano seguimiento de trayectorias irregulares arbitrarias mediante computadora. Tesis de Grado. Universidad Autónoma del Estado de Hidalgo. 2007.

[4] Oropeza, J. Suárez, S. Algoritmos y Métodos para el Reconocimiento de Voz en Español Mediante Sílabas. Computación y sistemas. Vol, 9. Núm 3. Pp. 270-286. 2006.

[5] Kischnin, A. Automatic Speech Recognition with the Parallel Cascade Neural Network. PhD Thesis, Tokyo, Japan. 1998.

[6] Jackson, L. Digital Filters and Signal Processing. Kluwer Academic Publishers. University of Loisville, Department of Electrical and Computer Engineering. U.S.A. 1986.

[7] Kosko, B. Neural Network for Signal Processing. Prentice Hall. U.S.A. 1992

[8] Sydral, A. Bennet, R. Greenspan, S. Applied Speech Technology. CRS Press. U.S.A. 1995.

[9] Cooley, J. Tukey, J. An algorithm for the machine calculation of complex fourier series. Math. Computat. 1965.

[10] Pose, M. Introducción a las Redes de Neuronas Artificiales. Departamento de Tecnologías de la Información y las Comunicaciones. Universidad da Coruña.

[11] Cooley, J. Lewis, P. Welch, P. Historical notes on the fast fourier transform. IEEE Transactions on Audio and Electroacoustics, 15:2608211; 262, June 1967.

[12] Rossum, G. Drake, F. El tutorial de Python. Python Software Foundation, Septiembre, 2009.

[13] Schaul, T. Bayer, J. Wierstra, D. Yi, S. Felder, M. Sehnke, F. Rückstieß, T. Schmidhuber, J. PyBrain. Journal of Machine Learning Research, 2010.

[14] Beyreuther, M. Barsch, R. Krischer, L. Megies, T. Behr, Y. Wassermann, J. (2010), ObsPy: A Python Toolbox for Seismology, SRL, 81(3), 530-533

[15] Krischer, L. Megies, T. Barsch, R. Beyreuther, R. Lecocq, T. Caudron, C. Wassermann, J. (2015), ObsPy: a bridge for seismology into the scientific Python ecosystem, Computational Science & Discovery, 8(1), 014003

[16] Pérez, Y. Osuna, I. Ibarra, R. Villegas, J. Reconocimiento de Voz usando Redes Neuronales para el Control de una Silla de Ruedas. IX Semana Nacional de Ingeniería Electrónica. Avances en Ingeniería Electrónica. Universidad Autónoma Metropolitana Azcapotzalco 2013.