UNIVERSIDAD TECNOLÓGICA EQUINOCCIAL

FACULTAD DE CIENCIAS DE LA INGENIERÍA

PROTOTIPO DE ROBOT MANIPULADOR CON VISIÓN ARTIFICIAL

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO

DE INGENIERO MECATRONICO

NOMBRE: GABRIEL RICARDO BORJA MOSCOSO

DIRECTOR: DANIEL MIDEROS

Quito, Septiembre 2012

DERECHOS DE AUTOR

DECLARACIÓN DE AUTORÍA

DECLARACIÓN

Yo Gabriel Ricardo Borja Moscoso, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Universidad Tecnológica Equinoccial puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____

Gabriel Ricardo Borja Moscoso

CERTIFICACIÓN DE AUTORÍA

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Gabriel Ricardo Borja Moscoso, bajo mi dirección y supervisión.

_____

Daniel Mideros

DIRECTOR DELTRABAJO

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

ÍNDICE DE FIGURAS

RESUMEN

En el presente trabajo se realizo un estudio y desarrollo de un prototipo de robot manipulador con visión artificial, el cual se elaborara con materia prima de PVC para la armadura o mecanismo del robot es decir sus brazos y sus base, para la parte electrónica se desarrollo un sistema que es capaz de transmitir en forma serial datos para el control de los servomotores que utilizamos, los cuales tienen un torque de 15Kg por centímetro, suficiente para manejar el peso del PVC. Los microcontroladores utilizados son el 16f628A para controlar cada servomotor y 16F877A como cerebro principal para el funcionamiento del sistema. El objetivo del robot manipulador es diferenciar dos tipos de piezas, y clasificarlas por su color.

El sistema de control se desarrollara en Matlab, con una interface GUI la cual nos da las opciones necesarias para realizar la separación de las piezas.

En el momento de hacer las pruebas se encontró que de cada 5 pruebas con 5 piezas 3 negras y 2 blancas los resultados fueron 100% exactos, es decir cada pieza se la llevo al lugar donde tenían que estar.

En la realización de otra prueba de 5 intentos con 6 piezas 3 negras y 3 blancas con medidas de 3cm de diámetro, los resultados fueron 100% exactos. Pero en el momento que el diámetro de las piezas bajo de

1cm a 2cm se encontraron problemas en el momento de encontrar el centro de cada pieza y por esto no llegaron a su destino, para solucionar este problema se aumento el tamaño del efector final que es un electroimán, y con esto se logro recoger todas las piezas.

SUMMARY

Was designed and built a prototype robot manipulator with vision, is made from PVC material for armor or robot mechanism that is his arms and base for electronics was developed a system that is capable of transmitting on a serial data for controlling the servo motors that use, which have a torque of 15kg, sufficient to handle the weight of PVC, and microcontrollers are used to control each actuator 16F628A 16F877A as main brain and for the operation of the system. He came to see it working in a real environment, is in practice that reached its goal is to distinguish two types of parts, and reach the position where they must be placed, with favorable results for a prototype.

At the time of testing was found in 5 trials with 5 rooms 3 black and 2 white the results were 100% accurate, is each piece took her to the place where they belonged.

In another test of 5 attempts with 6 pieces 3 black and 3 white with measures 3cmin diameter, the results were 100% accurate. But the moment that the diameter of the pieces of 1cm to 2cm in problem found in time to find the center of each piece and thus never reached their destination, to solve this problem is increasing the size of the end effect or is an electromagnet, and with this I collect all the pieces.

1. INTRODUCCIÓN

Para el diseño y construcción de un prototipo de robot manipulador hay que tomar en cuenta principalmente los problemas en su desarrollo.

Uno de los problemas clásicos en el diseño de brazos mecánicos es su peso, el cual genera un torque (fuerza angular) que debe ser soportado por motores, que a su vez tienen que tener fuerza adicional para soportar el torque generado por el peso de un objeto que pueda sujetar.

Para ello se utilizara como materia prima los tubos de PVC, ya que es un material resistente y liviano.

Como objetivos principales para el desarrollo del prototipo de robot manipulador con visión artificial encontramos:

- Analizar la bibliografía del tema para solucionar el problema.
- Diseñar los modelos electrónicos, mecánicos, y lógicos del robot manipulador.
- Establecer las normas y procedimientos de control que guiaran la relación entre el usuario y el robot y entre PC y el robot manipulador.
- Diseñar y construir las partes mecánicas como rodamientos brazos y bases
- Diseñar y construir las tarjetas electrónicas, y efectores finales del robot manipulador.
- Diseñar la programación de control y de reconocimiento con visión artificial.

Realizando la investigación apropiada, sobre la mecánica, electrónica y diseño de control para un prototipo de robot manipulador con visión artificial, llegamos a establecer que el presente proyecto beneficiará a los laboratorios ya que se estima percibir una ayuda en la experimentación, entrenamiento, practica y desarrollo de las materias en las cuales el estudiante debe

interactuar con un robot, y así lograr saber si lo que se ha realizado en la programación está cumpliendo y se podrá observar en un medio físico que es el robot.

Con el desarrollo de un Robot Manipulador incorporado Visión Artificial, para las practicas de robótica en la carrera de mecatronica, se podrá solucionar problemas de control operario maquina y así familiarizar a los estudiantes como trabajar con una maquina a nivel HMI como se trabaja en las industrias.

2. MARCO TEORICO

2.1 INTELIGENCIA ARTIFICIAL

La inteligencia artificial puede tener varias definiciones citadas por distintos investigadores, como los siguientes:

Según Charniak y McDermott, (1985). “La inteligencia artificial es el estudio de las facultades mentales mediante el uso de modelos computacionales”.

Según Winston, (1992). “La inteligencia artificial es el estudio de los cálculos que permiten percibir, razonar y actuar”.

Según Luger y Sttubbledfield, (1993). “La inteligencia artificial es la rama de la ciencia de la computación que se ocupa de la automatización de la conducta inteligente”.

Tomando en cuenta estos conceptos, se llega a entender que la inteligencia artificial es un proceso matemático e informático para simular el funcionamiento de la inteligencia humana, se deberá primero conocer que es la inteligencia humana o abordar algunas de sus propiedades, las cuales nos permitirán tener información para continuar con la inteligencia artificial.

2.1.1 Propiedades de la inteligencia humana

Algunos ejemplos de las propiedades de la inteligencia humanas, que son necesarias para el desarrollo de la inteligencia artificial, son las siguientes:

- Habilidad de enfrentar nuevas situaciones
- Habilidad de resolver problemas
- Habilidad de responder preguntas
- Habilidad para elaborar planes

La inteligencia humana aun es un concepto estudiado por biólogos sicólogos y filósofos de nuestra generación, por ser un tema extenso y de mucho estudio, no nos enfocaremos en el mismo pero conociendo algunas características de la inteligencia humana, comenzaremos el estudio de la inteligencia artificial.

2.1.2 Antecedentes de la inteligencia artificial (IA)

Ponce Cruz, (2010) considera que uno de los primeros pasos hacia la IA fueron dados hace muchos años por Aristóteles (384-322 A.C), cuando se dispuso a codificar y tratar de explicar ciertos temas del razonamiento deductivo que el llamo silogismos. Otro intento de inteligencia FUE ARS MAGNA un sistema capaz de responder cualquier pregunta su diseño era de ruedas. George Boole (Boole 1854) comenzó a desarrollar los fundamentos de la lógica proporcional, Boole con esto quería recoger algunos indicios probables sobre la naturaleza y la constitución de la mente humana.

M. C. Ricardo Fuentes Covarrubias, (2008). La historia de la Inteligencia Artificial se remonta a los primeros años de la década de los 40, con trabajos teóricos fundamentalmente dado el estado de la Informática en ese momento, cuando los matemáticos Warren McCullock y Walter Pitts, específicamente en 1943, desarrollan los algoritmos matemáticos necesarios para posibilitar el trabajo de clasificación, o funcionamiento en sentido general, de una red neuronal. En 1949 Donald Hebb desarrolló un algoritmo de aprendizaje para dichas redes neuronales creando, conjuntamente con los trabajos de McCullock y Pitts, la escuela coneccionista. Esta escuela se considera actualmente como el origen de lo que hoy se conoce como Inteligencia Artificial (IA).

Herbert Simon, el físico Allen Newell, uno de los padres de la IA actual, y J.C. Shaw, programador de la RAND Corp. y compañero de Newell,

desarrollan el primer lenguaje de programación orientado a la resolución de problemas de la IA, el IPL-11. Un año más tarde estos tres científicos desarrollan el primer programa de IA al que llamaron "Logic Theorist", el cual era capaz de demostrar teoremas matemáticos. Este programa demostró 38 de los 52 teoremas, del segundo capítulo de "Principia Mathematica" de Russel y Whitehead.

En el mismo año surge la IA como disciplina de la Informática en la Conferencia de Computación de Dartmouth. En dicha conferencia se estableció como conclusión fundamental la posibilidad de simular inteligencia humana en una máquina.

Funcionamiento de la inteligencia se lo puede explicar de la siguiente manera:

"Nuestro cerebro no posee un acceso directo al mundo exterior. Solamente podemos operar sobre una representación interna la cual se corresponde con una colección de estructuras de símbolos. Dichas estructuras pueden tomar la forma de un patrón físico cualquiera, por ejemplo un vector de interruptores eléctricos dentro de un ordenador, o un conjunto de neuronas activadas en un cerebro biológico. Un sistema inteligente (cerebro u ordenador) puede operar sobre las estructuras con el objetivo de transformarlas en otra construcción. El pensamiento consiste en la extensión, o desarrollo, de estas estructuras, descomponiéndolas y reformándolas, destruyendo algunas y creando nuevas.

La inteligencia entonces no constituye nada más que la habilidad de procesar estructuras de símbolos. Existe en un entorno diferente al hardware que le da soporte, lo trasciende y puede tomar diferentes formas físicas."

Desde este punto se desarrollaron sistemas de comprensión de preguntas basados en inteligencia artificial como:

En 1961 se desarrolla SAINT (Simbolic Automatic INTegrator) por James Slagle el cual se orienta a la demostración simbólica en el área del álgebra.

En 1964 Bertrand Raphael construye el sistema SIR (Semantic Information Retrieval) el cual era capaz de comprender oraciones en inglés. Es decir que se le hacien diferentes preguntas y el sistema de inteligencia artificial sabia como interpretarlas y responder de igual manera en una oración que era entendible para el operador.

En los primeros años de la década del 60 Frank Rosemblatt desarrolla, en la Universidad de Cornell, un modelo de la mente humana a través de una red neuronal y produce un primer resultado al cual llama Perceptron.

Este sistema era una extensión del modelo matemático concebido por McCullock y Pitts para las neuronas, y funcionaba basándose en el principio de "disparar" o activar neuronas a partir de un valor de entrada el cual modifica un peso asociado a la neurona, si el peso resultante sobrepasa un cierto umbral la neurona se dispara y pasa la señal a aquellas con las que está conectada. Al final, en la última capa de neuronas, aquellas que se activen definirán un patrón el cual sirve para clasificar la entrada inicial.

Este trabajo constituye la base de las redes neuronales de hoy en día.

2.1.3 Ramas de la inteligencia artificial

Algunas de las ramas de la inteligencia artificial son:

- Lógica difusa
- Redes neurales artificiales
- Algoritmos genéticos

2.2 LÓGICA DIFUSA

Ponce Cruz, (2010) considera que la lógica difusa tiene sus raíces en la teoría de conjuntos difusos desarrollada por Zadeh en la década de los 60, la que propone que un elemento siempre pertenece en un cierto grado a un conjunto y nunca pertenece del todo al mismo, esto permite establecer una

manera eficiente para trabajar con incertezas, así como para acondicionar el conocimiento en forma de reglas hacia un plano cuantitativo, factible de ser procesado por computadores.

Toda lógica consiste en formalizar el pensamiento humano, desde este punto de vista.

La lógica difusa se basa en una regla que no tienen límites discretos, si no que se prologan en un conjunto, permitiendo manejar de mejor manera la ambigüedad. Esto es muy útil ya que es como tienden a pensar las personas, en términos relativos no absolutos.

La lógica difusa es llevar a un sistema entender un lenguaje natural, ya que no maneja términos precisos como por ejemplo hace frio, hace calor o el precio es alto, es decir que se encuentran en un rango variable.

Hay que dejar claro los términos sistemas difusos y probabilidad, los 2 operan sobre el mismo rango numérico pero los conceptos son diferentes.

Hay que saber que la diferencia semántica es importante. La frase probabilística supone que hay una probabilidad de conocer la categoría en que se encuentra un objeto. Los grados difusos no son lo mismo que probabilidades. Las probabilidades miden si algo va ocurrir o no, la lógica difusa en cambio mide el grado en el cual algo ocurre o si alguna condición existe.

La lógica difusa se ha empleado con bastante éxito en la industria, principalmente en Japón, y cada vez se está usando en gran multitud de campos. La primera vez que se usó de forma importante fue en el metro japonés, con excelentes resultados. A continuación se citan algunos ejemplos de su aplicación:

Sistemas de control de acondicionadores de aire,

Sistemas de foco automático en cámaras fotográficas,

Electrodomésticos familiares (frigoríficos, lavadoras...),

Optimización de sistemas de control industriales,

Sistemas de escritura,

Mejora en la eficiencia del uso de combustible en motores,

Sistemas expertos del conocimiento (simular el comportamiento de un experto humano),

Tecnología informática.

Bases de datos difusas: Almacenar y consultar información imprecisa. Para este punto, por ejemplo, existe el lenguaje FSQL.

Lógica difusa VS lógica clásica

Lógica clásica:

Establece que cualquier enunciado o proposición puede tener un valor lógico verdadero o falso, en definitiva 1 y 0.

De esta forma es posible desarrollar toda una lógica basada en leyes de este tipo.

Para dar un ejemplo claro de lo que es la lógica clásica y sus diferencias entre la lógica difusa se ha escogido el funcionamiento de una puerta.

Según la lógica clásica la puerta solo puede tener 2 estados, cerrada (0) o abierta (1).

Lógica difusa:

Establece que una puerta no tiene por qué estar necesariamente abierta o cerrada, existen además otros estados, por ejemplo.

- Puerta abierta (1)
- Puerta bastante abierta (0.8)

- Puerta abierta a medias (0.5)
- Puerta casi cerrada (0.1)
- Una puerta está cerrada (0)

Además de ello la lógica difusa se trabajo con graficas las cuales nos permiten entender el funcionamiento de la misma.

En la **Figura 1**. podemos observar el grafico de lógica clásica el cual nos indica que su valor solo es 1 o 0, y en el caso del grafico de lógica difusa tenemos un grafico de una curva el cual nos indica los valores que puede tomar desde que la puerta está cerrada hasta cuando la puerta está abierta.

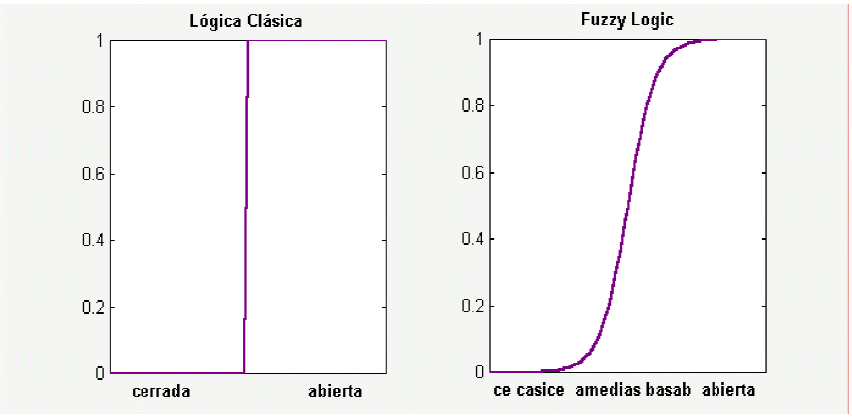


Figura 1. Imagen de lógica clásica y lógica difusa de una puerta Tomada del libro Inteligencia Artificial (Pedro Ponce, 2011, 1ª edición, Inteligencia artificial con aplicaciones a La Ingeniería)

Ventajas de Lógica Difusa

- La principal ventaja de utilizar términos lingüísticos como:

A medias, bastante, casi, un poco, mucho, algo, etc, está en que permite plantear el problema en los mismos términos en los que lo haría un experto humano.

- El éxito de esta técnica radica en que “El mundo es Fuzzy”.

En otras palabras, no tiene sentido buscar la solución a un problema no perfectamente definido por medio de un planteamiento matemático muy exacto, cuando es el ser humano el primero que razona empleando la inexactitud.

2.2.1 Proceso de Fuzzificación

Ponce Cruz, (2010) considera que el proceso de fuzzificación consiste en convertir una variable real en un grado de pertenencia que cuantifica el grado de posesión hacia su correspondiente variable lingüística.

- Las variables lingüísticas son representativas de situaciones como:

Positivo, alrededor de, alto, medio, etc.

- El primer paso consiste en tomar las entradas y determinar el grado al que ellos pertenecen a cada uno de los conjuntos difusos apropiados.
- La entrada siempre es un valor numérico limitado al universo del discurso de la variable de entrada (0-10).

2.2.2 Funciones de pertenencia

Las funciones de pertenencia representan las coordenadas difusas del atributo. Son funciones continuas, que pueden ser básicamente de los tipos:

- Trapezoidales y Triangulares: Son funciones lineales por tramos, pero representan una discontinuidad en la primera derivada que hereda la acción de control.
- Exponenciales: (distribución normal), muestran un comportamiento muy adecuado y no representan discontinuidad en la derivada.
- Polinómicas: Son funciones sencillas de calcular y tienen una forma similar a la de las funciones de densidad normal.

En la **Figura 2.** Se puede observar el grafico de las funciones de pertenencia.

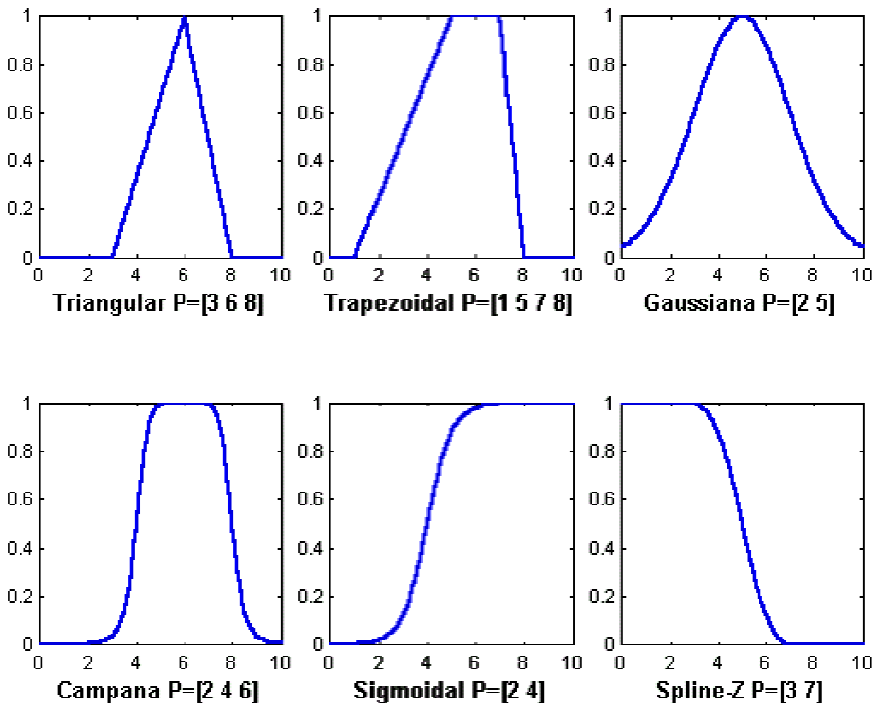


Figura 2. Imagen función de pertenencia, tomada del libro Inteligencia Artificial (Pedro Ponce, 2011, 1ª edición, Inteligencia artificial con aplicaciones a La Ingeniería)

2.2.3 Operadores difusos

Cuando una variable cubre el dominio de más de una variable lingüística, la variable difusa final es inferida por alguna operación que toma en cuenta el grado de pertenencia de cada una de las variables.

Los operadores más comunes son:

Operación Max(): asigna la correspondiente al valor máximo Operación Min(): asigna la correspondiente al valor mínimo que son equivalentes a las sentencias “or y “and” de la lógica booleana.

En la **Tabla 1**. Se puede observar los operadores difusos y sus resultados.

TABLA 1 Operadores difusos

A	B	A and B	A or B
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

2.2.4 Definición de las reglas

Ponce Cruz, (2010), considera que la segunda etapa de la lógica difusa es cuando se realizan o se proponen las reglas lingüísticas (inferencia) que servirán de guía para que el sistema se comporte de mejor manera para alcanzar los objetivos del usuario, el grado de pertenencia se evalúa junto a las reglas.

Ejemplo de cómo poner reglas.

Enunciados de reglas:

1. Si (servicio es pobre) o (comida es mala) entonces (restaurant es barato)

2. Si (servicio es bueno) entonces (restaurant es promedio)

3. Si (servicio es excelente) o (comida es rica) entonces (restaurant es caro)

Método de agregación de reglas:

La agregación es cuando se unifican las salidas de cada regla en forma paralela.

Posteriormente se realiza la defuzzificación del resultado.

2.2.5 Defuzzificación

El proceso inverso llamado defuzzificación transforma un conjunto difuso, es decir un conjunto de variables lingüísticas con sus respectivos grados de pertenencia, en un número real.

 El método más común es asimilarlo al centro de gravedad de la combinación de cada una de las reglas inferidas.

 También se usa el criterio máximo, que escoge el punto donde la función inferida tiene su máximo o el criterio de la media de los máximos.

2.3 REDES NEURONALES ARTIFICIALES (RNA)

Ponce Cruz, (2010), considera que la tecnología neural trata de reproducir el funcionamiento del cerebro y la resolución de problemas. Así como los humanos aprenden de los problemas que han resuelto, una red neural tomo como ejemplo problemas que ya se han resuelto para construir un sistema que toma decisiones y realiza clasificaciones. Los problemas adecuados para la aplicación neural son aquellos que no tienen solución computacional precisa o que tienen muchos algoritmos complejos como el caso de la identificación y tratamiento de imágenes.

Las redes neuronales artificiales (RNA) son aplicadas en diversos ámbitos de la actividad humana. Una de sus aplicaciones es como herramienta de análisis de información, específicamente dentro de la Bibliometría.

Una red neuronal artificial (RNA) es un esquema de computación distribuida inspirada en la estructura del sistema nervioso de los seres humanos. La arquitectura de una red neuronal es formada conectando múltiples procesadores elementales, siendo éste un sistema adaptivo que posee un algoritmo para ajustar sus pesos (parámetros libres) para alcanzar los

requerimientos de desempeño del problema basado en muestras representativas.

Por lo tanto podemos señalar que una RNA es un sistema de computación distribuida caracterizada por:

- Un conjunto de unidades elementales, cada una de las cuales posee bajas capacidades de procesamiento.
- Una densa estructura interconectada usando enlaces ponderados.
- Parámetros libres que deben ser ajustados para satisfacer los requerimientos de desempeño.
- Un alto grado de paralelismo.

Es importante señalar que la propiedad más importante de las redes neuronales artificiales es su capacidad de aprender a partir de un conjunto de patrones de entrenamientos, es decir, es capaz de encontrar un modelo que ajuste los datos. El proceso de aprendizaje también conocido como entrenamiento de la red puede ser supervisado o no supervisado.

El aprendizaje supervisado consiste en entrenar la red a partir de un conjunto de datos o patrones de entrenamiento compuesto por patrones de entrada y salida. El objetivo del algoritmo de aprendizaje es ajustar los pesos de la red w de manera tal que la salida generada por la RNA sea lo más cercanamente posible a la verdadera salida dada una cierta entrada. Es decir, la red neuronal trata de encontrar un modelo al procesos desconocido que generó la salida y. Este aprendizaje se llama supervisado pues se conoce el patrón de salida el cual hace el papel de supervisor de la red.

En cambio en el aprendizaje no supervisado se presenta sólo un conjunto de patrones a la RNA, y el objetivo del algoritmo de aprendizaje es ajustar los pesos de la red de manera tal que la red encuentre alguna estructura o configuración presente en los datos.

McCulloch and Pitts, (1943) concibieron un modelo abstracto y simple de una neurona artificial, este es el elemento básico de procesamiento en una red neuronal artificial.

En la **Figura 3.** Se observa el modelo que está compuesto por un vector de pesos w_= (w1,..., wd) ^T equivalente a las conexiones sinápticas en una neurona real, w0 es el umbral de acción o activación, el vector x es la entrada y el escalar y la salida de la unidad. La actividad consiste en generar una única salida y a partir de la aplicación de la función de activación γ a la suma ponderada entre el vector de entrada X_= (x1 ,..., xm) ^T y el vector de pesos w_= (w1,..., wd) ^T más un sesgo w0, obteniéndose la siguiente expresión en la **ecuación** [1 1].

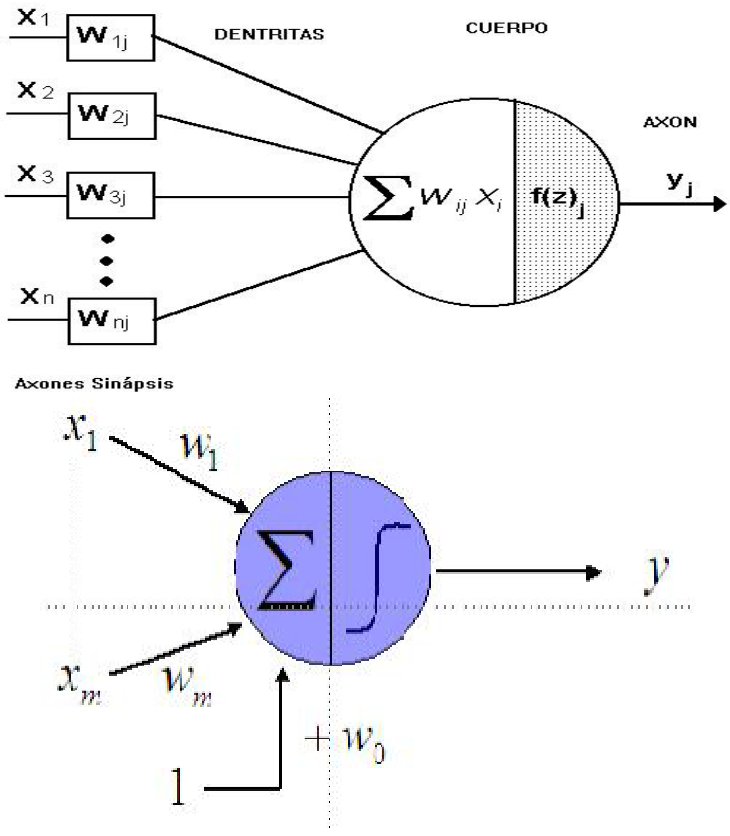


Figura 3. Esquema de una neurona artificial

http://investigacionsimulacion.blogspot.com/2010/11/representacion-del-conocimiento.html

$$y = u(\sum_{i=1}^{m} w_i x_i + w_o)$$

Ecuación [2 1]

Donde γ es una función no-lineal. La función propuesta por McCulloch-Pitts posee una salida binaria ±1 conocida como la función de todo o nada que equivale a la función signo dada por:

Otra función con salida binaria es la función escalón unitario descrita por:

$$y(z) = U(z) = \begin{cases} 1z & \geq 0 \\ 0z & < 0 \end{cases}$$ **Ecuación** [2 2]

Cuando se consideran neuronas con respuestas de procesamiento gradual, entonces se pueden usar funciones de activación de forma lineal γ(z)=z o de forma sigmoidal como la función logística:

$$y(z) = \frac{1}{1+e^{-z}}$$ **Ecuación** [2 3]

O la tangente hiperbólica de **y (z)=tanh (z).**

2.3.1 Ventajas de redes neuronales artificiales (RNA)

Ponce Cruz, (2010), considera que las (RNA) tienen muchas ventajas debido a que están basadas en la estructura del sistema nervioso, principalmente el cerebro.

- Aprendizaje: Las RNA tienen la habilidad de aprender mediante una etapa que se llama etapa de aprendizaje. Esta consiste en proporcionar a la RNA datos como entrada a su vez que se le indica cuál es la salida (respuesta) esperada.

- Auto organización: Una RNA crea su propia representación de la información en su interior, descargando al usuario de esto.
- Tolerancia a fallos: Debido a que una RNA almacena la información de forma redundante, ésta puede seguir respondiendo de manera aceptable aun si se daña parcialmente.
- Flexibilidad: Una RNA puede manejar cambios no importantes en la información de entrada, como señales con ruido u otros cambios en la entrada (por ejemplo si la información de entrada es la imagen de un objeto, la respuesta correspondiente no sufre cambios si la imagen cambia un poco su brillo o el objeto cambia ligeramente).
- Tiempo real: La estructura de una RNA es paralela, por lo cual si esto es implementado con computadoras o en dispositivos electrónicos especiales, se pueden obtener respuestas en tiempo real.

2.3.2 Clasificación de redes neuronales artificiales (RNA)

Una primera clasificación de las redes de neuronas artificiales que se suele hacer es en función del patrón de conexiones que presenta. Así se definen tres tipos básicos de redes:

Dos tipos de redes de propagación hacia delante o acíclicas en las que todas las señales van desde la capa de entrada hacia la salida sin existir ciclos, ni conexiones entre neuronas de la misma capa.

Monocapa: Ejemplos: perceptrón, Adaline.

Multicapa: Ejemplos: perceptrón multicapa.

Las redes recurrentes que presentan al menos un ciclo cerrado de activación neuronal. Ejemplos: Elman, Hopfield, máquina de Boltzmann.

Aprendizaje: Una segunda clasificación que se suele hacer es en función del tipo de aprendizaje de que es capaz (si necesita o no un conjunto de

entrenamiento supervisado). Para cada tipo de aprendizaje encontramos varios modelos propuestos por diferentes autores:

Aprendizaje supervisado: necesitan un conjunto de datos de entrada previamente clasificado o cuya respuesta objetivo se conoce. Ejemplos de este tipo de redes son: el perceptrón simple, la red Adaline, el perceptrón multicapa, red backpropagation, y la memoria asociativa bidireccional.

Aprendizaje no supervisado o autoorganizado: no necesitan de tal conjunto previo. Ejemplos de este tipo de redes son: las memorias asociativas, las redes de Hopfield, la máquina de Boltzmann y la máquina de Cauchy, las redes de aprendizaje competitivo, las redes de Kohonen o mapas autoorganizados y las redes de resonancia adaptativa (ART).

Redes híbridas: son un enfoque mixto en el que se utiliza una función de mejora para facilitar la convergencia. Un ejemplo de este último tipo son las redes de base radial.

Aprendizaje reforzado: se sitúa a medio camino entre el supervisado y el autoorganizado.

Tipo de entrada: Finalmente también se pueden clasificar las RNAs según sean capaces de procesar información de distinto tipo en:

Redes analógicas: procesan datos de entrada con valores continuos y, habitualmente, acotados. Ejemplos de este tipo de redes son: Hopfield, Kohonen y las redes de aprendizaje competitivo.

Redes discretas: procesan datos de entrada de naturaleza discreta; habitualmente valores lógicos booleanos. Ejemplos de este segundo tipo de redes son: las máquinas de Boltzmann y Cauchy, y la red discreta de Hopfield.

2.3.3 Elementos básicos que componen una red neuronal.

En la **Figura 4**. Se puede observa la conexión de una red neuronal artificial totalmente conectada y distinguir sus capas.

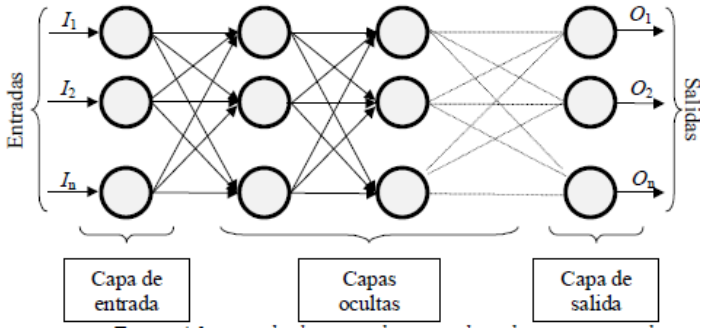


Figura 4. Red neuronal totalmente conectada, tomada del libro Inteligencia Artificial (Pedro Ponce, 2011, 1ª edición, Inteligencia artificial con aplicaciones a La Ingeniería).

La misma está constituida por neuronas interconectadas y arregladas en tres capas (esto último puede variar). Los datos ingresan por medio de la "capa de entrada", pasan a través de la "capa oculta" y salen por la "capa de salida". Cabe mencionar que la capa oculta puede estar constituida por varias capas. Antes de comenzar el estudio sobre las redes neuronales, se debe aprender algo sobre las neuronas y de cómo ellas son utilizadas por una red neuronal. En la **Figura 5**. Se compara una neurona biológica con una neurona artificial. En la misma se pueden observar las similitudes entre ambas (tienen entradas, utilizan pesos y generan salidas).

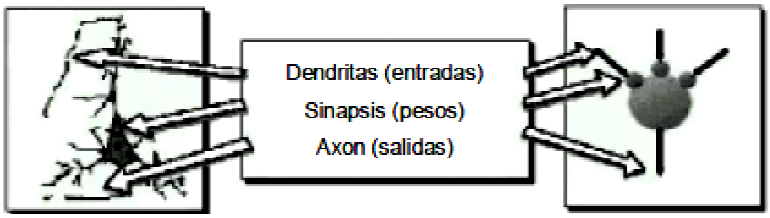


Figura 5. Comparación entre una neurona biológica y una artificial, tomada

http://itsce-redesneuronales.blogspot.com/2011/05/elementos-basicos-que-componen-un-red.html

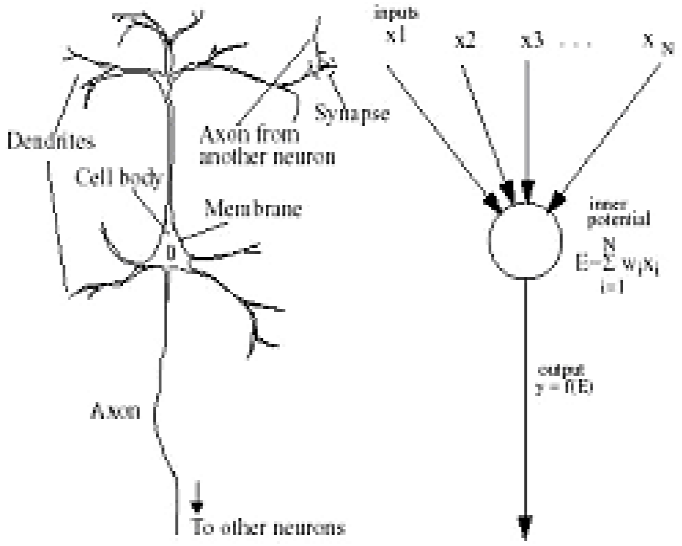


Figura 6. Neurona real y neurona artificial, tomada de http://campusvirtual.unex.es/cala/epistemowikia/images/1/13

2.3.4 Aplicaciones de una red neuronal.

Las redes neuronales pueden utilizarse en un gran número y variedad de aplicaciones, tanto comerciales, en medicina, militares.

Neuronal Networks: Basics and Applications, (1998) considera que se pueden desarrollar redes neuronales en un periodo de tiempo razonable, con la capacidad de realizar tareas concretas mejor que otras tecnologías. Cuando se implementan mediante hardware (redes neuronales en chips VLSI), presentan una alta tolerancia a fallos del sistema y proporcionan un alto grado de paralelismo en el procesamiento de datos. Esto posibilita la inserción de redes neuronales de bajo coste en sistemas existentes y recientemente desarrollados.

Hay muchos tipos diferentes de redes neuronales; cada uno de los cuales tiene una aplicación particular más apropiada. Algunas aplicaciones comerciales son:

Biología:

- Aprender más acerca del cerebro y otros sistemas.

- Obtención de modelos de la retina.

Empresa:

- Evaluación de probabilidad de formaciones geológicas y petrolíferas.

- Identificación de candidatos para posiciones específicas.

- Explotación de bases de datos.

- Optimización de plazas y horarios en líneas de vuelo.

-Optimización del flujo del tránsito controlando convenientemente la temporización de los semáforos.

- Reconocimiento de caracteres escritos.

- Modelado de sistemas para automatización y control.

Medio ambiente:

- Analizar tendencias y patrones.

- Previsión del tiempo.

Finanzas:

- Previsión de la evolución de los precios.

- Valoración del riesgo de los créditos.

- Identificación de falsificaciones.

- Interpretación de firmas.

Manufacturación:

- Robots automatizados y sistemas de control (visión artificial y sensores de presión, temperatura, gas, etc.).

- Control de producción en líneas de procesos.

- Inspección de la calidad.

Medicina:

- Analizadores del habla para ayudar en la audición de sordos profundos.

-Diagnóstico y tratamiento a partir de síntomas y/o de datos analíticos (electrocardiograma, encefalogramas, análisis sanguíneo, etc.).

- Monitorización en cirugías.

- Predicción de reacciones adversas en los medicamentos.

- Entendimiento de la causa de los ataques cardíacos.

Militares:

- Clasificación de las señales de radar.

- Creación de armas inteligentes.

- Optimización del uso de recursos escasos.

- Reconocimiento y seguimiento en el tiro al blanco.

La mayoría de estas aplicaciones consisten en realizar un reconocimiento de patrones, como ser: buscar un patrón en una serie de ejemplos, clasificar patrones, completar una señal a partir de valores parciales o reconstruir el patrón correcto partiendo de uno distorsionado. Sin embargo, está creciendo el uso de redes neuronales en distintos tipos de sistemas de control.

Desde el punto de vista de los casos de aplicación, la ventaja de las redes neuronales reside en el procesado paralelo, adaptativo y no lineal.

El dominio de aplicación de las redes neuronales también se lo puede clasificar de la siguiente forma: asociación y clasificación, regeneración de patrones, regresión y generalización, y optimización.

2.3.5 Algoritmo genético.

Ponce Cruz, (2010), considera que los Algoritmos Geneticos (AGs) son métodos adaptativos que pueden usarse para resolver problemas de

búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin (1859).

Por imitación de este proceso, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas.

Los Algoritmos Genéticos usan una análoga directa con el comportamiento natural. Trabajan con una población de individuos, cada uno de los cuales representa una solución factible a un problema dado. A cada individuo se le asigna un valor o puntuación, relacionado con la bondad de dicha solución. En la naturaleza esto equivaldrá al grado de efectividad de un organismo para competir por unos determinados recursos. Cuanto mayor sea la adaptación de un individuo al problema, mayor será la probabilidad de que el mismo sea seleccionado para reproducirse, cruzando su material genético con otro individuo seleccionado de igual forma. Este cruce producirá nuevos individuos {descendientes de los anteriores {los cuales comparten algunas de las características de sus padres.

Cuanto menor sea la adaptación de un individuo, menor será la probabilidad de que dicho individuo sea seleccionado para la reproducción, y por tanto de que su material genético se propague en sucesivas generaciones.

De esta manera se produce una nueva población de posibles soluciones, la cual reemplaza a la anterior y verifica la interesante propiedad de que contiene una mayor proporción de buenas características en comparación con la población anterior. A lo largo de las generaciones las buenas características se propagan a través de la población. Favoreciendo el cruce de los individuos mejor adaptados, van siendo exploradas las aéreas más prometedoras del espacio de búsqueda. Si el Algoritmo Genético ha sido bien diseñado, la población convergerá hacia una solución óptima del problema.

El poder de los Algoritmos Genéticos proviene del hecho de que se trata de una técnica robusta, y pueden tratar con éxito una gran variedad de problemas provenientes de diferentes aéreas, incluyendo aquellos en los que otros métodos encuentran dificultades. Si bien no se garantiza que el

Algoritmo Genético encuentre la solución optima del problema.

2.4 VISION ARTIFICIAL Y TRATAMIENTO DE IMAGENES.

G.A. Baxes, (1994), R.C. González y R.E. Woods, (1993). Consideran que La visión artificial tiene como finalidad la extracción de información del mundo físico a partir de imágenes, utilizando para ello un computador.

Un sistema de Visión Artificial actúa sobre una representación de una realidad, que le proporciona información sobre brillo, colores, formas, etc. Estas representaciones suelen estar en forma de imágenes estáticas, escenas tridimensionales o imágenes en movimiento.

2.4.1 Etapas de un sistema de visión artificial

El ser humano captura la luz con los ojos, y esta información circula a través del nervio óptico hasta el cerebro donde se procesa.

Existen razones para creer que el primer paso de este procesado consiste en encontrar elementos más simples en los que descomponer la imagen (como segmentos y arcos). Después el cerebro interpreta la escena y por último actúa en consecuencia. La visión artificial, en un intento de reproducir este comportamiento, define tradicionalmente cuatro fases principales:

• La primera fase, que es puramente sensorial, consiste en la captura o adquisición de las imágenes digitales mediante algún tipo de sensor.

• La segunda etapa consiste en el tratamiento digital de las imágenes, con objeto de facilitar las etapas posteriores. En esta etapa de procesamiento previo es donde, mediante filtros y transformaciones geométricas, se eliminan partes indeseables de la imagen o se realzan partes interesantes de la misma.

• La siguiente fase se conoce como segmentación, y consiste en aislar los elementos que interesan de una escena para comprenderla.

• Por último se llega a la etapa de reconocimiento o clasificación. En ella se pretende distinguir los objetos segmentados, gracias al análisis de ciertas características que se establecen previamente para diferenciarlos.

Estas cuatro fases no se siguen siempre de manera secuencial, sino que en ocasiones deben realimentarse hacia atrás. Así, es normal volver a la etapa de segmentación si falla la etapa de reconocimiento, o a la de pre proceso, o incluso a la de captura, cuando falla alguna de las siguientes.

En la Figura 7. Se observa las etapas del proceso de visión artificial

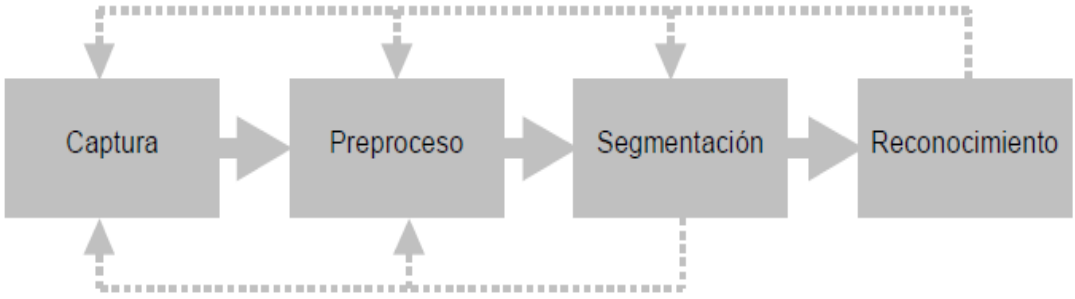


Figura 7. Etapas de visión artificial, tomada de Libro Visión por computadora (Vélez Serrano, Visión por Computador, 2ª Edición, 2007)

2.4.2 Elemento para realizar un sistema de visión artificial.

Incluyendo hardware específico para acelerar ciertas operaciones, los elementos imprescindibles son:

• Un sensor óptico para captar la imagen: Una cámara de vídeo, una cámara fotográfica, una cámara digital, un escáner... uniéndole un conversor analógico-digital cuando sea preciso.

• Un computador que almacene las imágenes y que ejecute los algoritmos de preprocesado, segmentación y reconocimiento de la misma.

2.4.3 Captura

Las imágenes digitales son “señales” discretas, que suelen tener origen en una “señal” continua. Por ejemplo, una cámara digital toma imágenes del mundo real que es continuo (tanto el espacio, como el espectro de la radiación reflejada por los objetos se consideran continuos);.

En el proceso de obtención de imágenes digitales se distinguen dos etapas.

La primera, captura, utiliza un dispositivo generalmente óptico, con el que obtiene información relativa a una escena. En la segunda digitalización, transforma esa información, que es una señal con una o varias componentes continuas, en la imagen digital, que es una señal con todas sus componentes discretas.

Para capturar una imagen se suele distinguir entre dispositivos pasivos (basados generalmente en el principio de cámara oscura) y dispositivos activos (basados en el escaneo). Esta clasificación no incluye todas las formas posibles de creación de imágenes, como por ejemplo la construcción de imágenes sintéticas.

La cámara oscura se puede usar para capturar imágenes de escenas tridimensionales (del mundo real) y proyectarlas en un plano bidimensional. Dispositivos de este tipo son las cámaras fotográficas y las cámaras de vídeo. Este modelo además se puede usar para capturar imágenes de elementos bidimensionales, como fotografías y documentos, como por ejemplo hacen los escáneres de cámara. También se pueden usar dos o

más cámaras para capturar diferentes perspectivas de una misma escena y construir una representación 3D de la misma.

Este esquema es fundamentalmente distinto del basado en cámara, ya que existe un elemento activo (generalmente un haz de luz láser) que recorre la escena que se desea capturar. Por tanto son imprescindibles dos dispositivos, uno emisor del haz de luz y otro el receptor. El escáner emite el haz de luz y éste, tras chocar con la imagen que se escanea, es recogido en el detector de luz. Repitiendo este proceso de manera continua se puede construir una señal que corresponde a una representación de la escena.

Los dispositivos basados en el escaneo también se usan con diferentes fines. Así, los escáneres-láser pueden capturar escenas 3D directamente, y los escáneres de tambor permiten capturar imágenes de elementos bidimensionales.

Los dispositivos basados en cámara aventajan a los basados en escaneo en velocidad. Además son más simples y se parecen más al sistema visual humano. Es de prever que, con el tiempo, los modelos de cámara terminen superando también a los de escaneo en cuanto a calidad de la imagen obtenida, ya que su principal cuello de botella, que se encuentra actualmente en el elemento digitalizador, parece que puede ser mejorado sensiblemente con nuevos desarrollos tecnológicos.

2.4.4 La digitalización

Es el proceso de paso del mundo continuo (o analógico) al mundo discreto (o digital). En la digitalización normalmente se distinguen dos procesos: el muestreo (“sampling”) y la cuantización (“quantization”).

2.4.5 Muestreo

El muestreo de una señal continua consiste en la medición a intervalos (discretización) respecto de alguna variable (generalmente el tiempo o el espacio), siendo su parámetro fundamental la frecuencia de muestreo, que representa el número de veces que se mide un valor analógico por unidad de cambio.

Mediante el muestreo se convierte una imagen IC, que es algo continuo, en una matriz discreta ID de N×M píxeles. El número de muestras por unidad de espacio sobre el objeto original conduce al concepto de resolución espacial de la imagen. Ésta se define como la distancia, sobre el objeto original, entre dos píxeles adyacentes. Sin embargo la unidad de medida de resolución espacial más habitual suele ser los píxeles por pulgada (comúnmente DPIs9) siempre medidos sobre el objeto original.

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

$$I_{c(x,y)\in\mathcal{R}\ en\ donde\ x,y\in\mathcal{R}}$$

Ecuación [2. 5]

Al formato:

$$I_{D(x,y)\in\mathcal{R}\ en\ donde\ x,y\in\ N\ y\ 0\leq x\leq N-1,0\leq y\leq M-1}$$

Ecuación [2. 6]

Que se puede representar en forma matricial:

$$I_{D(x,y)}=\begin{cases} I_{D(0,0)} & I_{D(0,1)} & I_{D(0,M-1)} \\ I_{D(1,0)} & I_{D(1,1)} & I_{D(1,M-1)} \\ I_{D(N-1,0)} & I_{D(N-1,1)} & I_{D(N-1,M-1)} \end{cases}$$

Ecuación [2. 7]

2.4.6 Cuantización

La segunda operación es la cuantización de la señal, que consiste en la discretización de los posibles valores de cada píxel. Los niveles de cuantización suelen ser potencias de 2 para facilitar el almacenamiento en el computador de las imágenes, ya que éstos utilizan el byte10 como unidad mínima de memoria directamente direccionable. Así, suelen usarse 2, 4, 16 ó 256 niveles posibles. De esta forma, ID que pertenece a N se convierte en IDC (discreta cuantizada) que pertenece a N. El número de niveles posibles define la resolución radiométrica.

$$I_{DC}(x, y) \in N \qquad \textbf{Ecuación } [2.\ 8]$$

Donde:

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

$$0 \leq I_{DC}(x, y) \leq 2^q - 1 \qquad \textbf{Ecuación } [2.\ 9]$$

En la **Figura 8.** se ve un cuadro del uso de los pixeles en una imagen tomando en cuenta que cuando las imágenes sólo tienen información sobre el brillo se habla de imágenes en niveles de gris y se suelen utilizar hasta 256 niveles para representar los tonos intermedios desde el negro (0) hasta el blanco (255). Si sólo se permiten dos niveles de cuantización (normalmente blanco y negro) se habla de imágenes bitonales o imágenes binarias. Para el caso del color suelen usarse 256 niveles para representar la intensidad de cada uno de los tres colores primarios (RGB). De esta forma se obtienen 16 millones de colores aproximadamente (256x256x256) y se habla de imágenes en color real. En algunos casos puede necesitarse mayor resolución radiométrica y se usan 4096 niveles por banda de color en vez de 256, o incluso más.

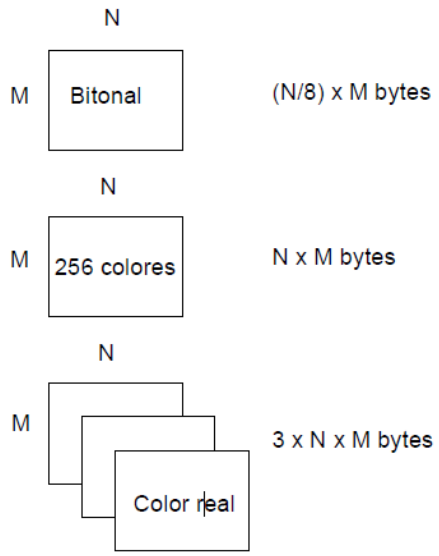


Figura 8. Imágenes digitales y tamaño en bits, tomada de Libro Visión por computadora (Vélez Serrano, Visión por Computador, 2ª Edición, 2007)

El proceso de digitalización requiere evaluar qué resolución espacial y qué resolución radiométrica se precisan para representar adecuadamente una imagen.

Dicho de otra forma, con qué frecuencia se muestrean los píxeles (frecuencia de muestreo), y qué gama de colores se permite (elección de la paleta).

También hay que tener en cuenta que dependiendo del uso que se vaya a hacer de una imagen, la elección de los parámetros de digitalización puede variar de una forma menos objetiva. Así, para la publicación de un periódico en blanco y negro, 16 niveles de intensidad podrían ser suficientes, pero elegir menos de 80 por 80 píxeles por pulgada de resolución espacial sería inadmisible; mientras que para una imagen, con vistas a su reconocimiento, aunque podría ser preciso utilizar más niveles de intensidad, se podría permitir una resolución espacial menor.

Se ha hablado de un muestreo de nivel uniforme pero existen tipos de muestreos que no son uniformes, El muestreo no uniforme consiste en el uso de diferente frecuencia de muestreo para diferentes zonas de la imagen.

De esta forma, las zonas más interesantes pueden tener una resolución espacial mayor que las menos interesantes, consiguiendo un ahorro de los recursos del sistema.

La cuantización no uniforme se basa en el uso de paletas. Una paleta consiste en un conjunto de colores a los que se les asigna una referencia. Los píxeles de las imágenes que usan paletas contienen como valor la referencia al color de la paleta que quieren presentar.

Representación de la imagen y estructuras de Datos

Las imágenes suelen almacenarse en los ordenadores en forma de ficheros. En este punto se analizarán las estructuras que se usan a tal efecto, los métodos utilizados para optimizar el espacio requerido y algunos de los diferentes formatos estándar (TIFF, GIF, BMP, JPG...).

Formatos de representación

Existen multitud de formatos de ficheros de imágenes de tipo mapa de bits. Se puede hablar de ficheros tipo BMP, TIFF, GIF, JFIF, PGM... Cada uno frece ciertas ventajas que otros formatos pueden no contemplar. La **Tabla 2.** recoge las principales características de algunos de estos formatos.

En la **Figura 9.** se ve 2 imágenes de distinto formato tomando en cuenta que la imagen .jpg tiene mejor resolución que la .png.

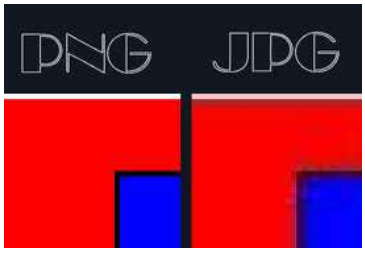


Figura 9. Imagen de dos tipos distintos de imágenes, tomada de http://vagabundia.blogspot.com/2008/08/los-formatos-de-las-imgenes.html

TABLA 2. Ficheros gráficos y características principales

Formato	Color Real	Paleta	Grises	Bitonal	Compresión	Origen	Multi-Imagen
Bitmap	SI	SI	SI	SI	Run-Length	Windows	NO
TIFF	SI	SI	SI	SI	JPG, LZW, Runs, CCITT4, CCITT3, PackBits	Estándar	SI
JFIF	SI	NO	SI	NO	JPEG	Estándar	NO
JPG2000	SI	NO	SI	NO	JPEG 2000	Estándar	NO
PCX	NO	SI	NO	NO	Propia	Windows	NO
PGM	NO	NO	SI	NO	NO	Unix	NO
GIF	NO	SI	SI	SI	LZW	Estándar	SI

Tomada de, http://www.desarrolloweb.com/articulos/19.php

Relación entre pixeles

Dependiendo de la situación de los píxeles y de los valores que tienen, se definen ciertas relaciones de vecindad y conectividad.

Para todo punto p de coordenadas (x, y) se dice que un píxel q pertenece a sus 4- vecinos y se escribe **qEN4(p)** si y sólo si q tiene coordenadas:

(x-1,y) ó (x,y-1) ó (x+1,y) ó (x,y+1)

Para todo punto p de coordenadas (x, y) se dice que un píxel q pertenece a sus 8-vecinos y se escribe **qEN8(p)** si y sólo si q tiene coordenadas:

(x-1,y) ó (x,y-1) ó (x+1,y) ó (x,y+1) ó (x-1,y-1) ó (x-1,y+1) ó (x+1,y-1) ó (x+1,y+1)

En la **Figura 10.** se observa cómo se produce un efecto vecindad alrededor del punto p.

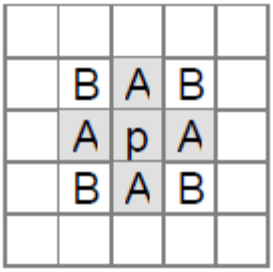


Figura 10. Los 4-vecinos de p son los puntos A. Los 8-vecinos de p son los puntos A y B.

Se ha visto que una imagen se asimila a una matriz cada uno de cuyos elementos es un píxel. Entre los píxeles de esta matriz se puede definir una relación que define dos píxeles como conectados cuando son vecinos y sus valores son similares desde algún punto de vista. Formalmente, se define un conjunto V que representa los valores compatibles para que dos píxeles que sean vecinos se diga que están conectados:

V = {Valores de los píxeles que definen conectividad}

Tras este estudio se comprende la importancia de la adecuada elección de los dispositivos de captura, su configuración y el formato de representación en un sistema informático. Se ha visto que esta elección debe depender del uso que se vaya a dar a la información capturada.

Operaciones aritmético-lógicas entre pixeles

Estas operaciones son, con diferencia, las más usadas a cualquier nivel en un sistema de tratamiento de imágenes, ya que son las que se utilizan para leer y dar valores a los píxeles de las imágenes. Las operaciones básicas son:

• Conjunción.- Operación lógica AND entre los bits de dos imágenes. Se usa para borrar píxeles en una imagen.

• Disyunción.- Operación lógica OR entre los bits de dos imágenes. Se usa para añadir píxeles a una imagen.

• Negación.- Inversión de los bits que forman una imagen. Se usa para obtener el negativo de una imagen.

• Suma.- Suma de los valores de los píxeles de dos imágenes.

• Resta.- Resta de los valores de los píxeles de dos imágenes.

• Multiplicación.- Multiplicación de los valores de los píxeles de una imagen por los de otra. Se usa para añadir textura a una imagen.

• División.- División de los valores de los píxeles de una imagen entre los de otra.

Se ha visto que en imágenes en niveles de gris se suele utilizar el valor 255 para representar el blanco y el 0 para el negro. Así, la operación de conjunción entre negro y blanco da como resultado negro. Si para el negro se utilizase 255 y para el blanco 0 los resultados de las operaciones conjunción y disyunción estarían intercambiados. En la **Figura 11.** se pueden apreciar los resultados de diferentes operaciones sobre las imágenes en niveles de gris A y B. Sobre imágenes en color los resultados serían similares.

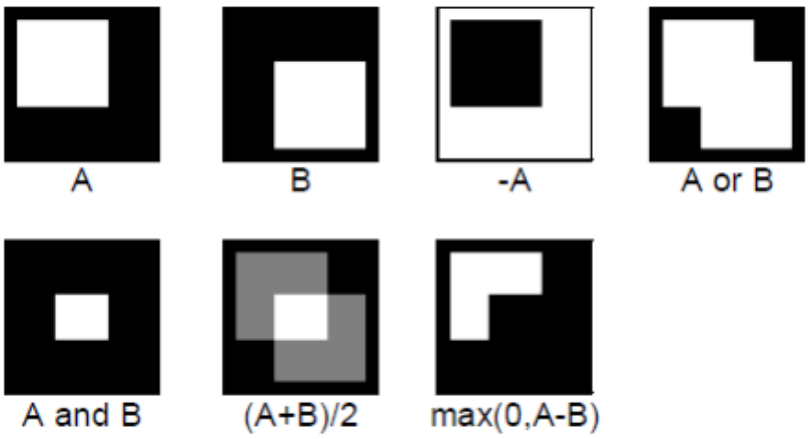


Figura 11.- Ejemplos de operaciones aritméticas y lógicas. Los píxeles a negro corresponden a bits a 0, los blancos a bits a 255, tomada de Libro Visión por computadora (Vélez Serrano, Visión por Computador, 2ª Edición, 2007)

Cuando se realiza operaciones aritméticas se debe tener la precaución de verificar que el resultado R de una operación cae dentro del dominio de valores permitidos. En caso contrario se puede dividir el valor R por un factor que consiga que el resultado pertenezca al dominio deseado. Si se desea que los valores no salgan de un rango [m ... M] se puede ajustar el resultado usando las funciones máximo y mínimo de dos valores. Para ello se toma como resultado definitivo el máximo entre R y el valor mínimo permitido m y luego el mínimo con el máximo permitido M, consiguiendo de esta forma que los valores nunca salgan del rango especificado.

Al implementar algoritmos que realizan operaciones aritmético-lógicas es una buena idea utilizar múltiplos del tamaño de la palabra del procesador en las transacciones con memoria, a fin de minimizar el número de operaciones de acceso a la misma. También se debe implementar los algoritmos de forma tal que en accesos consecutivos se referencien posiciones de memoria cercanas, de manera que el criterio de proximidad referencial, usado por las memorias caché, permita un rendimiento óptimo.

Operaciones sobre el histograma

Se conoce como histograma de los niveles de cuantización de la imagen, un diagrama de barras en el que cada barra tiene una altura proporcional al número de píxeles que hay para un nivel de cuantización determinado. Habitualmente, en el eje de abscisas se disponen los diferentes niveles de cuantización de valores que pueden tomar los píxeles de tal imagen, mientras el eje de ordenadas refleja el número de píxeles que habrá para cada nivel de cuantización.

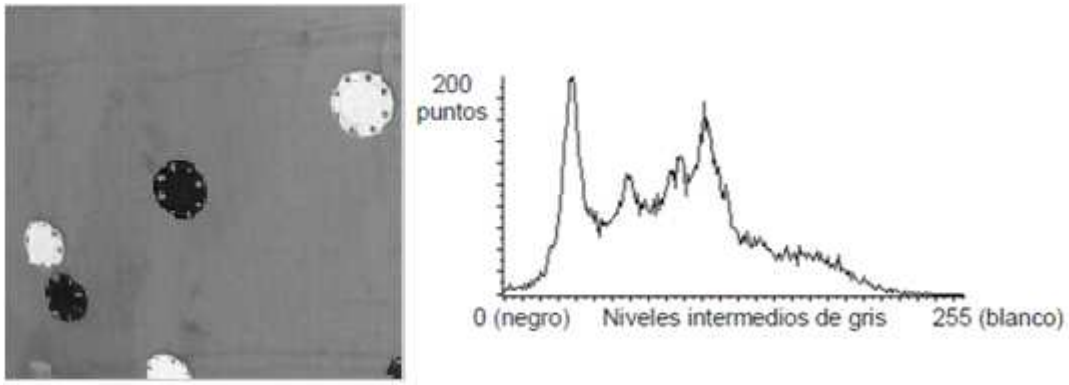


Figura 12. Imagen en niveles de gris y su correspondiente histograma, Tomada del sistema que se está desarrollando en Matlab.

Se debe notar que los histogramas no dicen nada sobre la disposición espacial de los píxeles. Por ello, un histograma es una forma de representación de imágenes en la que se produce pérdida de información. A partir del histograma de una imagen es imposible deducir la imagen que lo originó. Se deriva que aunque una imagen sólo puede tener un histograma, imágenes diferentes podrían tener el mismo histograma.

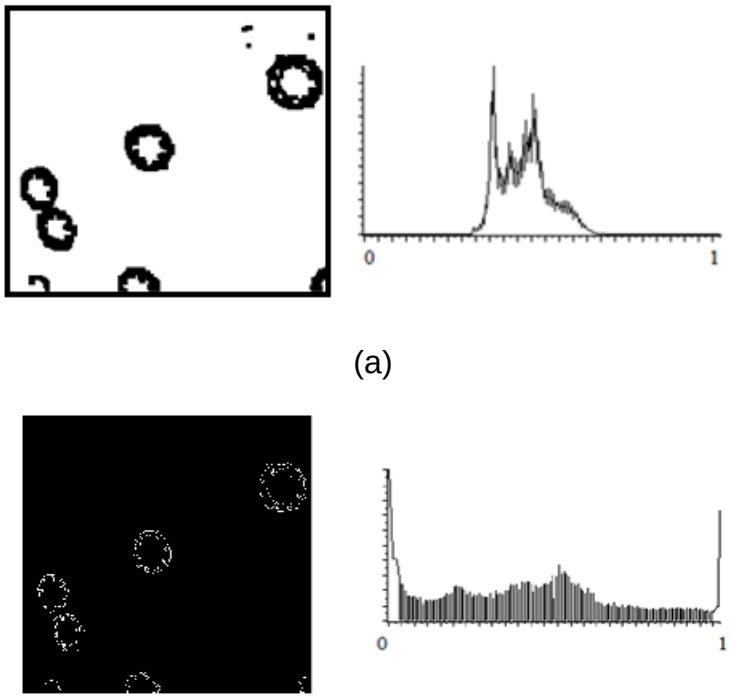
El histograma de una imagen en niveles de gris proporciona información sobre el número de píxeles que hay para cada nivel de intensidad (ver **Figura 12**).

En imágenes en color RGB se usan 3 histogramas, uno por cada componente de color. En el caso de imágenes de paleta, el histograma, si bien se puede calcular, tiene una utilidad menos evidente.

Al construir histogramas, para evitar la dependencia entre el número de píxeles o el número de niveles de cuantización y el tamaño del histograma, suelen normalizarse los ejes entre 0 y 1. Esta es la razón por la que en los ejes no suelen aparecer las unidades.

Se conoce como rango dinámico de una imagen al conjunto de todos los posibles valores de los píxeles que efectivamente se encuentran presentes en ella.

Cuando el rango dinámico de una imagen es pequeño, generalmente, se trata de imágenes con poco contraste, en las cuales se desaprovechan los recursos de captura. También puede darse el caso contrario, que ocurre cuando el histograma posee altos valores en los extremos de la escala, teniendo forma de “U”, en este caso se dice que la imagen está saturada.



(a)

(b)

Figura 13. (a) histograma de una imagen con poco contraste. (b) histograma de una imagen saturada, Tomada del sistema de visión artificial que se esta desarrollando.

El análisis del histograma de una imagen permite conocer detalles sobre la calidad de la misma y del proceso de captura que se ha utilizado para obtenerla.

Así, las imágenes consideradas de calidad suelen tener un rango dinámico amplio y no saturado. Las imágenes saturadas y las imágenes con un rango dinámico pequeño contienen menos información que las imágenes con un rango dinámico amplio y no saturado. No hace falta decir que para tareas de reconocimiento es mejor utilizar imágenes con mucha información. También

es importante para las tareas de reconocimiento que las imágenes tengan un alto nivel de contraste (sin llegar a estar saturadas), ya que esto implica que los detalles discriminantes se perciben con mayor claridad.

Cuando la calidad de una imagen digital es pobre, bien porque esté saturada, bien porque su rango dinámico sea pequeño, suele deberse a un mal ajuste del dispositivo de captura. Por ello, siempre que sea posible, la mejor opción consiste en repetir la captura y variar los parámetros del dispositivo de captura o las condiciones de iluminación de la escena. Debe recordarse, que aunque sobre una imagen digital se pueden realizar filtrados que visualmente produzcan efectos de aumento de contraste, disminución de la saturación o realzado de contornos, desde el punto de vista de la teoría de la información estos cambios sólo implican una pérdida de la información que contiene la imagen. Nunca un filtrado digital aumenta la información de una imagen, en el mejor de los casos no altera la información presente en la misma. Aún así, estos filtrados pueden resultar útiles para destacar elementos de la imagen que se necesiten en la etapa de reconocimiento.

Aumento y reducción de contraste

Las modificaciones del histograma se pueden visualizar eficazmente mediante las funciones de transferencia del histograma. Estas funciones corresponden a aplicaciones, pues para cada punto del dominio solo tiene un valor imagen. Estas aplicaciones están acotadas entre 0 y 1 tanto en la abscisa, que sé hacer corresponder con la entrada IE del filtro, como en la ordenada, que se corresponde con la salida IS del filtro.

En la **Figura 14.** se presentan tres ejemplos de funciones de transferencia: la función lineal, la función cuadrado y la función raíz cuadrada. La función de transferencia lineal (a) no introduce modificación alguna sobre el histograma, al coincidir exactamente los niveles de intensidad de la entrada y de la salida. La función cuadrado produce un oscurecimiento general de la imagen (en la figura (b) se aprecia que el rango entre 0 y 0’5 se hace

corresponder con el rango entre 0 y 0’25 que es más oscuro). Por último, la función raíz cuadrada (c) produce un aclarado general de la imagen.

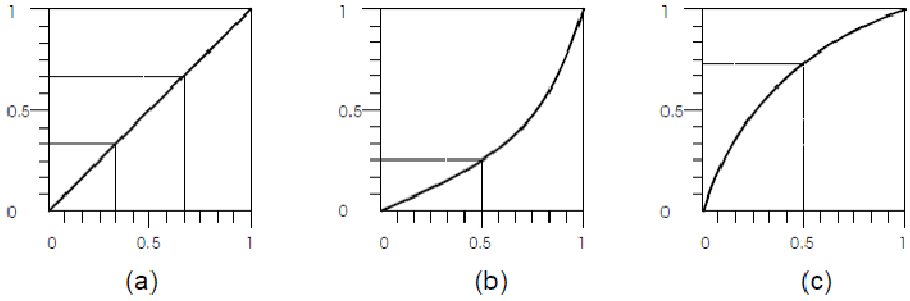


Figura 14. De izquierda a derecha las funciones lineal, cuadrado y raíz cuadrada, Tomada de Tratamiento de imágenes (Samira Hervella, 2011)

Se definió contraste como la diferencia de intensidad pronunciada. Ahora, en este contexto, se puede hablar de alto contraste en una imagen digital en niveles de gris si sobre el histograma se aprecia masas separadas.

En este contexto, un buen indicador del contraste de una imagen podría consistir en el cálculo de la desviación típica de su histograma.

Una función de transferencia que aclare los niveles claros y oscurezca los niveles oscuros, conseguirá sobre el conjunto de la imagen un efecto visual de aumento de contraste. Una función tal se puede obtener componiendo una función de transferencia del histograma que hasta el valor de 0’5 se comporte como la función cuadrado y que en adelante se comporte como la función raíz. En la **Figura 15.** Se ha representado esta función de transferencia. La función (b) de la misma figura produce el efecto contrario, esto es una disminución del contraste.

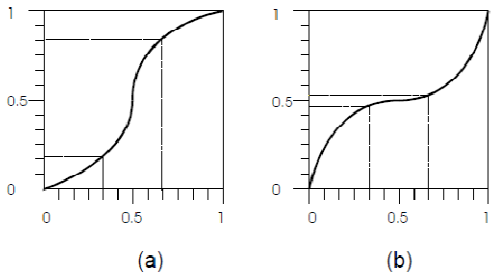
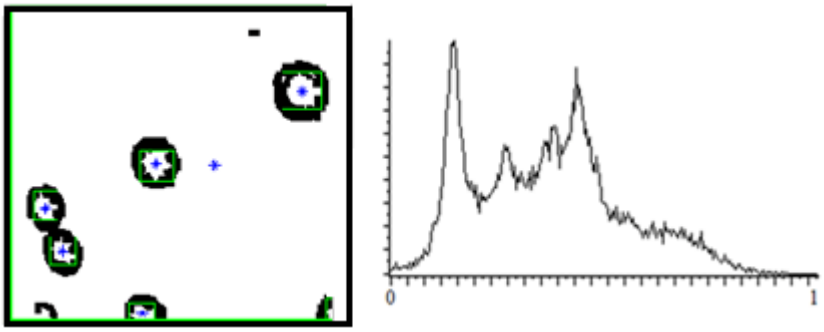


Figura 15. Funciones de transferencia para aumento y reducción de contraste, Tomada de Tratamiento de imágenes (Esther de Ves Cuenca, 2011)

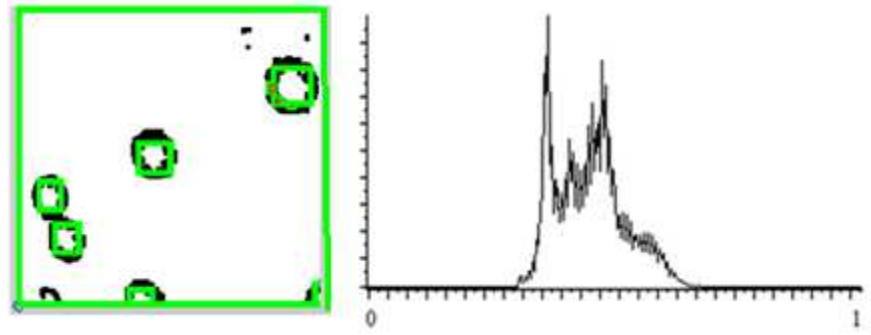
En general, una función de transferencia con una pendiente inferior a la unidad produce un efecto de reducción de contraste. Esto se debe a que concentra los valores de las intensidades de un rango R en un rango más pequeño R'. Por otro lado una función de transferencia con una pendiente superior a la unidad produce un efecto de aumento de contraste por razones inversas. Las imágenes de la **Figura 16.** en las que respectivamente se ha aumentado y reducido el contraste, han sido obtenidas aplicando estas funciones de transferencia. Así, al aplicar el filtro de aumento de contraste se aprecia en su histograma que se ha saturado los tonos claros y los oscuros, mientras que se ha reducido la densidad en la parte central del histograma. Esto quiere decir que, proporcionalmente, hay muchos más píxeles asociados a los tonos blancos y negros que al resto de tonos.

El filtro de reducción de contraste produce una imagen en la que se aprecia una reducción del rango dinámico del histograma. Esto también significa que la imagen contiene menos información que otra en la que el rango dinámico sea más amplio, ya que se ha homogeneizado zonas de la imagen que antes eran diferentes.

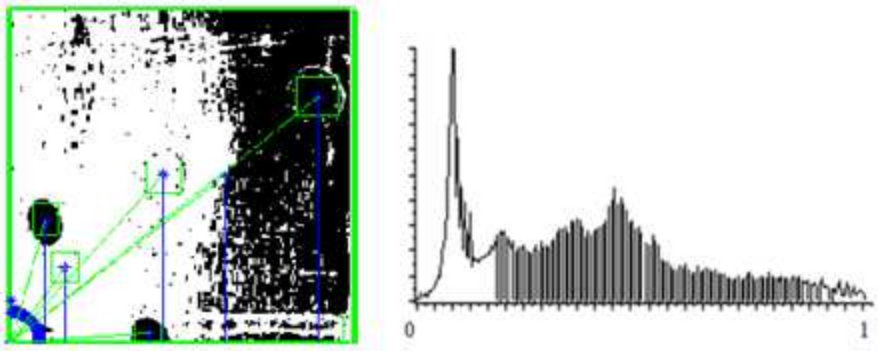
Aunque todas las transformaciones de histograma suelen expresarse matemáticamente con fórmulas que hay que aplicar sobre cada píxel (x, y), en la práctica, suelen tabularse. Es decir, se usa una tabla para acceder al valor correspondiente a la transformación en vez de realizar el cálculo cada vez. De esta forma, se aceleran las operaciones, más teniendo en cuenta el reducido número de niveles de intensidad involucrados (normalmente 256).



(a)



(b)



(c)

Figura 16. Transformaciones del histograma sobre la imagen: (a) imagen original con su correspondiente histograma; (b) resultado de una operación de disminución de contraste; (c) aumento de contraste, Tomada del sistema que se está desarrollando para visión artificial.

2.4.7 Segmentación

La segmentación es un proceso que consiste en dividir una imagen digital en regiones homogéneas con respecto a algunas características (como por ejemplo el brillo o el color) con el fin de facilitar un posterior análisis o reconocimiento automático. Localizar la cara de una persona dentro de la imagen de una fotografía o encontrar los límites de una palabra dentro de una imagen de un texto, constituyen ejemplos de problemas de

segmentación. En este capítulo se estudiarán diferentes enfoques para realizar el proceso de segmentación, aunque en la práctica se demuestra que la segmentación no tiene reglas estrictas a seguir, y dependiendo del problema en cuestión, puede ser necesario idear técnicas a medida.

También se tratará, al final del capítulo, el problema de la descripción de los objetos resultantes de la segmentación. Entonces, se estudiarán métodos que permiten una descripción de los objetos segmentados independiente de la posición y de la escala de los mismos.

La segmentación debe verse como un proceso que a partir de una imagen, produce otra en la que cada píxel tiene asociada una etiqueta distintiva del objeto al que pertenece. Así, una vez segmentada una imagen, se podría formar una lista de objetos consistentes en las agrupaciones de los píxeles que tengan la misma etiqueta.

La segmentación termina cuando los objetos extraídos de la imagen se corresponden unívocamente con las distintas regiones disjuntas a localizar en la misma. En este caso se habla de segmentación completa de la escena o imagen y en el caso contrario, de segmentación parcial. En una escena compleja, el resultado de la segmentación podría ser un conjunto de regiones homogéneas superpuestas y en este caso, la imagen parcialmente segmentada deberá ser sometida después a un tratamiento posterior con el fin de conseguir una segmentación completa.

El proceso de segmentación de una imagen depende del problema que se desee resolver. Por ejemplo, sobre una imagen de una página de texto se pueden segmentar las líneas de texto (si el objetivo es localizar la estructura de los párrafos), o las palabras y los caracteres que las forman (si se desea hacer OCR de los mismos), o los logotipos y membretes (si se desea clasificar el documento), etc. Por ello, dentro de una misma imagen pueden realizarse diferentes segmentaciones.

En general, el proceso de la segmentación suele resultar complejo debido, por un lado, a que no se tiene una información adecuada de los objetos a

extraer y, por otro, a que en la escena a segmentar aparece normalmente ruido. Es por esto que el uso de conocimiento sobre el tipo de imagen a segmentar o alguna otra información de alto nivel puede resultar muy útil para conseguir la segmentación de la imagen.

Algunos ejemplos típicos de procesos de segmentación son: tratar de separar los caracteres que forman una palabra dentro de una imagen de un texto, detectar ciertos tipos de células en imágenes médicas, extraer los vehículos que aparecen en una imagen de una carretera.

Los diferentes objetos que aparecen en una imagen pueden ser localizados atendiendo a aspectos como: sus contornos o su textura. Cada una de las técnicas que se estudiarán en este capítulo atienden a alguna de estas características, y para su estudio han sido englobadas en tres grupos: técnicas basadas en umbralización, basadas en detección de los contornos de los objetos y técnicas basadas en propiedades locales de las regiones.

2.5 LOS SISTEMAS MECÁNICOS

Constituyen la primera etapa para el desarrollo de un sistema mecatrónico. Estos sistemas requieren de un diseño y análisis que considere los componentes o actuadores eléctricos, neumáticos, hidráulicos, electrónicos, etc., para luego ser construidos.

Para desarrollar un sistema mecánico el primer paso es el diseño. En la actualidad los diseños son asistidos por programas informáticos conocidos con el nombre de sistemas CAD por sus siglas en inglés “Computer Aided Design”. El análisis del sistema puede realizarse por medio de un modelado matemático o de simulaciones en el computador.

Realizar un análisis del sistema tanto dinámico como cinemático es importante para dimensionar los actuadores que se requieren. El análisis cinemático permite encontrar las ecuaciones físicas del sistema para conocer la transmisión de movimientos (rotación y traslación) generados por

los actuadores. El análisis dinámico permite conocer los parámetros físicos como las velocidades, esfuerzos, aceleraciones, etc.; lo que permite establecer aspectos importantes para el modelo físico, tales como: tipo de material, rendimiento del mecanismo, potencia de los actuadores, etc.

Luego del diseño y análisis del sistema se determinan los materiales adecuados para su manufactura y los mecanismos adecuados para dar la forma del diseño, ya sea por medio de técnicas manuales, máquinas herramientas convencionales y procesos CAM “Computer Aided Manufacturing”. La tecnología CAM es una evolución de los procesos de manufactura que en la actualidad son cada vez más comunes por su versatilidad. Entre ellas están las máquinas de control numérico computarizado CNC o simple control numérico CN. Una vez construido cada uno de los elementos de sistema éstos se acoplan, garantizando la funcionalidad del sistema.

Cinemática

La descripción matemática del movimiento constituye el objeto de una parte de la física denominada cinemática.

Cinemática: Estudio del movimiento sin considerar las fuerzas que lo producen. Propiedades geométricas y temporales. Posición, velocidad, aceleración, derivadas superiores de la posición, etc.

Cinemática de los manipuladores: Propiedades geométricas y temporales del movimiento.

Los términos en un mecanismo:

Articulación: Conexión de dos cuerpos rígidos caracterizados por el movimiento de un sólido sobre otro.

Grado de libertad: Circular o prismático

Enlace: Cuerpo rígido que une dos ejes articulares adyacentes del manipulador. Posee muchos atributos: Peso, material, inercia, etc.

Movimiento

Se dice que un cuerpo se mueve cuando cambia su posición respecto de la de otros supuestos fijos, o que se toman como referencia. El movimiento es, por tanto, cambio de posición con el tiempo.

Es posible estudiar el movimiento de dos maneras:

a) describiéndolo, a partir de ciertas magnitudes físicas, a saber: posición, velocidad y aceleración (cinemática);

b) analizando las causas que originan dicho movimiento (dinámica).

Nos centraremos en la cinemática.

Y para ello cabe decir que, la cinemática es la parte de la física que estudia cómo se mueven los cuerpos sin pretender explicar las causas que originan dichos movimientos.

En cinemática directa: se conocen las variables articulares de una cadena de enlaces de un brazo articulado, su cálculo es sencillo (multiplicación matricial)

La cinemática directa determina la posición final del robot con respecto a un sistema de coordenada conocida.

En cinemática directa se conocen las variables articulares de una cadena de enlaces de un brazo articulado, su cálculo es sencillo (multiplicación matricial)

La cinemática directa determina la posición final del robot con respecto a un sistema de coordenada conocida. El sistema de coordenadas está definido por X; Y, Z.

Para el cálculo de la cinemática se definen cada una de las articulaciones y el ángulo referido con el sistema de coordenadas.

Para calcular la posición X, Y del brazo utilizando método geométrico se tienen las siguientes ecuaciones:

$$x1 = d1 \times sen(q1) + d2 \times sen(q1 + q2)$$ **Ecuación** [2.10]

$$y1 = d1 \times cos(q1) + d2 \times cos(q1 + q2)$$ **Ecuación** [2.11]

Cinemática inversa de un robot scara

La cinemática inversa determina la configuración que deben adoptar las articulaciones del robot para alcanzar una posición y orientación del extremo conocido.

Según Jacques Deanvit – Richard Hatemberg Las ecuaciones cinemáticas del brazo robot se encontraron utilizando el Algoritmo de Denavitt Hatenberg el cual plantea un método matricial que permite establecer de manera sistemática un sistema de coordenada para cada eslabón de una cadena articulada.

El método permite transformar un sistema de coordenadas de un vector referido a un sistema de coordenadas homogéneas de forma que:

$$T = \begin{Bmatrix} R_{3X3} & P_{3X1} \\ f_{1X3} & w_{1X1} \end{Bmatrix}$$ **Ecuación** [2.12]

Donde:

- R indica un vector de rotación
- P indica un vector de traslación
- f indica un vector de Perspectiva
- W indica un vector Escalado

De esta manera se puede establecer una matriz homogénea que contenga la información para cada uno de los sistemas de coordenadas:

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación [2.13]

Para la aplicación del método de Denavit Hartenberg se debe definir cada uno de los eslabones de un sistema articulado. Se asigna el sistema de coordenadas para cada uno y luego se aplica la matriz de transformación homogénea.

Una vez definido el sistema de coordenadas se debe considerar los parámetros de forma y tamaño de cada eslabón.

Para simplificar el desarrollo de cálculo se trabajó con un sistema informático, se ingresaron las ecuaciones para que el programa calcule la matriz homogénea y de esta forma obtener las ecuaciones para cada variable de posición en X, Y y Z.

2.5.1 Robótica.

Según Clive Gifford. La Robótica es la ciencia y la tecnología de los robots. Se ocupa del diseño, manufactura y aplicaciones de los robots. La robótica combina diversas disciplinas como son: la mecánica, la electrónica, la informática, la inteligencia artificial y la ingeniería de control. Otras áreas importantes en robótica son el álgebra, los autómatas programables y las máquinas de estados.

Por robot industrial de manipulación se entiende a una máquina de manipulación automática reprogramable y multifuncional con tres o más ejes que pueden posicionar y orientar materias, piezas, herramientas o dispositivos especiales para la ejecución de trabajos diversos en las

diferentes etapas de la producción industrial, ya sea en una posición fija o en movimiento.

2.5.2 Clasificación de robots.

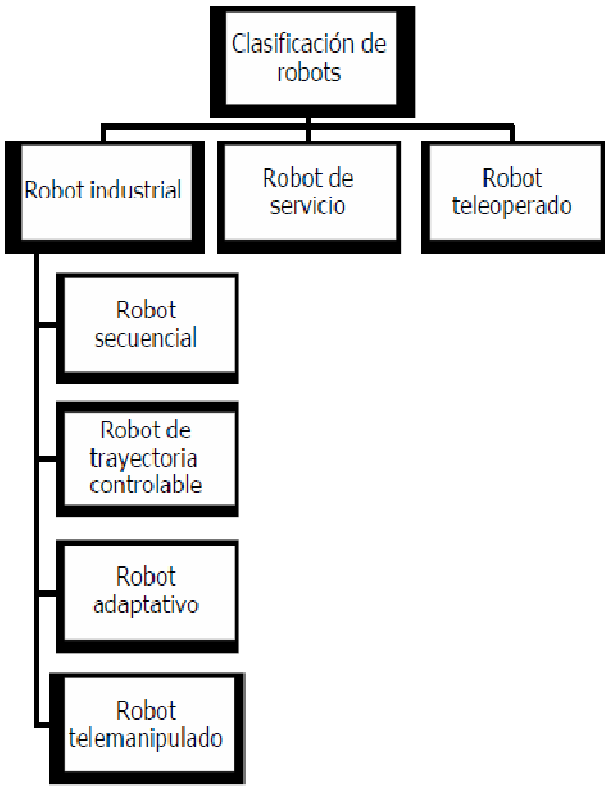


Figura 17. Clasificación de robots, tomada de http://robotec11.tripod.com/id4.html

2.5.3 Clasificación de Robots según su arquitectura

Los dispositivos y mecanismos que pueden agruparse bajo el concepto de Robots son muy diversos y, por lo tanto, es difícil establecer una clasificación coherente de los mismos que resista un análisis crítico y riguroso. La subdivisión de los Robots, con base en su arquitectura, se hace

en los siguientes grupos: Poli articulados, Móviles, Androides, Zoomórficos e Híbridos.

- Robots de muy diversa forma y configuración cuya característica es la de ser sedentarios y estar estructurados para mover sus elementos terminales en un determinado espacio de trabajo. Ejemplo: los cartesianos, industriales o manipuladores.
- Robots con grandes capacidades de desplazamiento, basados en carros o plataformas y dotados de un sistema locomotor de tipo rodante.
- Robots que intentan reproducir total o parcialmente la forma y el comportamiento cinemática del ser humano. Actualmente, los Androides son todavía dispositivos muy poco evolucionados y sin utilidad práctica, destinados especialmente a la experimentación. Un ejemplo de androide es el Asimo, fabricado por Toyota.
- Los Robots Zoomórficos constituyen una clase caracterizada principalmente por sus sistemas de locomoción que imitan a los diversos seres humanos. Éstos se agrupan en dos categorías: caminadores y no caminadores.
- Híbridos corresponden a aquellos de difícil clasificación, cuya estructura se sitúa la combinación con algunas de las anteriores ya expuestas, bien sea por conjunción o yuxtaposición. Por ejemplo, robots articulados y con ruedas (conjunción) o un cuerpo formado por un carro móvil y de un brazo semejante al de los robots industriales (yuxtaposición).

2.5.4 Estructura de un robot industrial:

Según Lewis, Clive Gifford. Un robot está formado por los siguientes elementos: estructura mecánica, transmisiones, actuadores, sensores, elementos terminales y controlador. Aunque los elementos empleados en los robots no son exclusivos de estos (máquinas herramientas y otras muchas máquinas emplean tecnologías semejantes), las altas prestaciones que se

exigen a los robots han motivado que en ellos se empleen elementos con características específicas. La constitución física de la mayor parte de los robots industriales guarda cierta similitud con la anatomía de las extremidades superiores del cuerpo humano, por lo que, en ocasiones, para hacer referencia a los distintos elementos que componen el robot, se usan términos como cintura, hombro, brazo, codo, muñeca, etc.

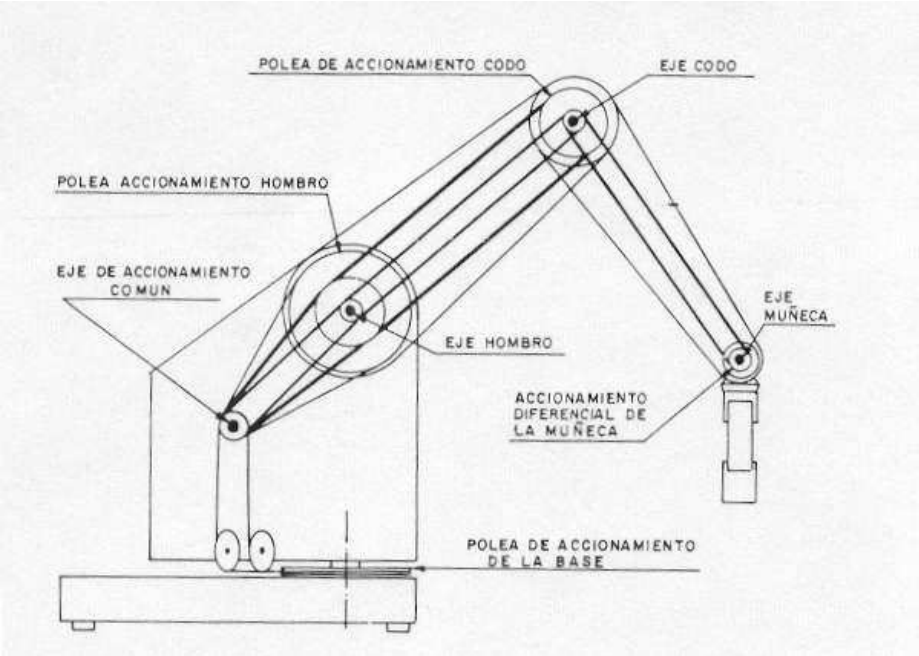


Figura 18. Partes simuladas de un robot manipulador, Tomada http://www.todorobot.com.ar/proyectos/brazo/brazo.htm

Los elementos que forman parte de la totalidad del robot son:

- Manipulador
- Controlador
- Dispositivos de entrada y salida de datos
- Dispositivos especiales

Mecánicamente, es el componente principal. Está formado por una serie de elementos estructurales sólidos o eslabones unidos mediante articulaciones que permiten un movimiento relativo entre cada dos eslabones consecutivos.

Las partes que conforman el manipulador reciben, entre otros, los nombres de: cuerpo, brazo, muñeca y actuador final (o elemento terminal). A este último se le conoce habitualmente como aprehensor, garra, pinza o gripper.

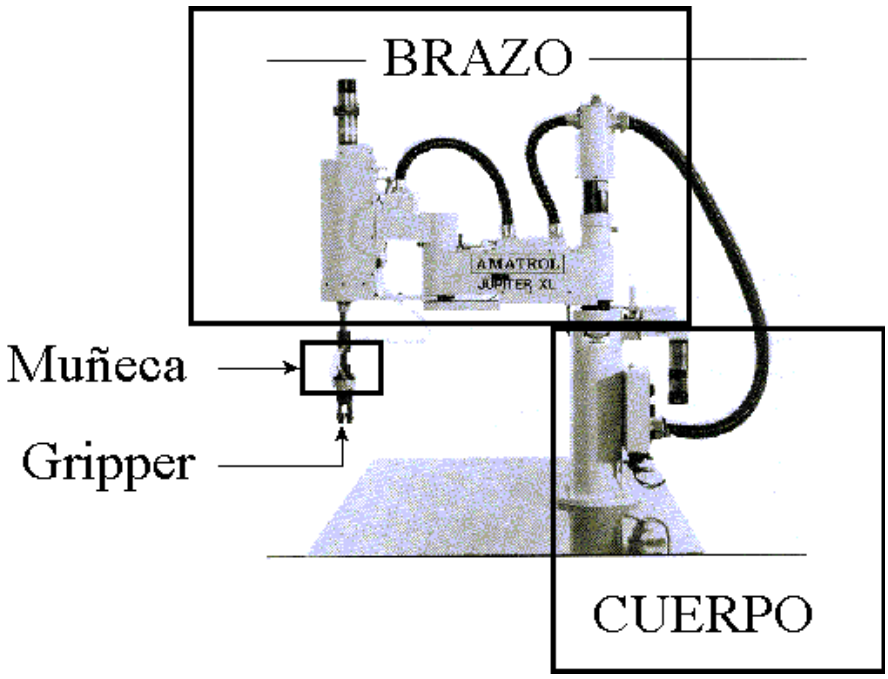
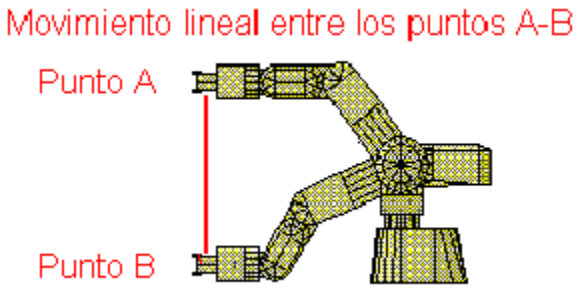


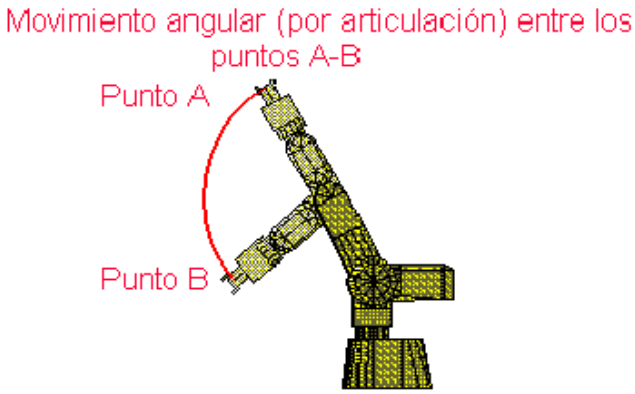
Figura 19. Partes robot manipulador tipo scara, tomada de http://cfievalladolid2.net/tecno/cyr_01/robotica/sistema/morfologia.htm

Cada articulación provee al robot de, al menos, un grado de libertad. En otras palabras, las articulaciones permiten al manipulador realizar movimientos:

Lineales que pueden ser horizontales o verticales.



Angulares (por articulación)



(En los dos casos la línea roja representa la trayectoria seguida por el robot).

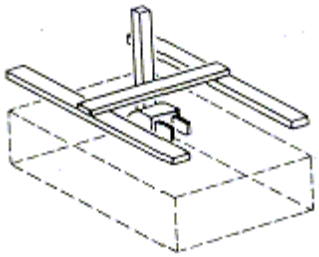
Existen dos tipos de articulación utilizados en las juntas del manipulador:

- Prismática /Lineal - junta en la que el eslabón se apoya en un deslizador lineal. Actúa linealmente mediante los tornillos sinfín de los motores, o los cilindros.
- Rotacional - junta giratoria a menudo manejada por los motores eléctricos y las transmisiones, o por los cilindros hidráulicos y palancas.

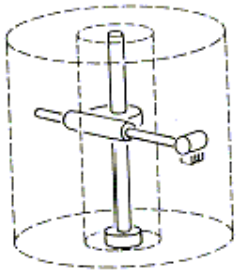
El actuador final (gripper) es un dispositivo que se une a la muñeca del brazo del robot con la finalidad de activarlo para la realización de una tarea específica. La razón por la que existen distintos tipos de elementos terminales es, precisamente, por las funciones que realizan. Los diversos

tipos podemos dividirlos en dos grandes categorías: pinzas y herramientas. Se denomina Punto de Centro de Herramienta (TCP, Tool Center Point) al punto focal de la pinza o herramienta. Por ejemplo, el TCP podría estar en la punta de una antorcha de la soldadura.

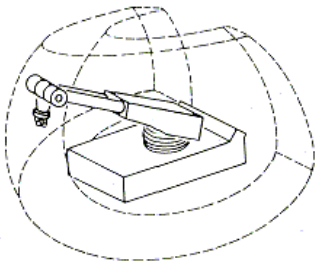
Para ilustrar lo que se conoce como volumen de trabajo regular y volumen de trabajo irregular, tomaremos como modelos varios robots.



a) El robot cartesiano y el robot cilíndrico presentan volúmenes de trabajo regulares. El robot cartesiano genera una figura cúbica.



b) El robot de configuración cilíndrica presenta un volumen de trabajo parecido a un cilindro (normalmente este robot no tiene una rotación de 360°)



c) Por su parte, los robots que poseen una configuración polar, los de brazo articulado y los modelos SCARA presentan un volumen de trabajo irregular.

Figura 20. Modelos de robots manipuladores, a), b), c), tomada de

http://www.um.es/docencia/barzana/IATS/lats09.html

3. METODOLOGIA Y MATERIALES

La metodología mecatronica es un diseño de ingeniería en un proceso complejo que implica interacción entre varias habilidades y disciplinas.

Diseño mecatrónico es: Concurrente, Sistema global integrado desde un inicio.

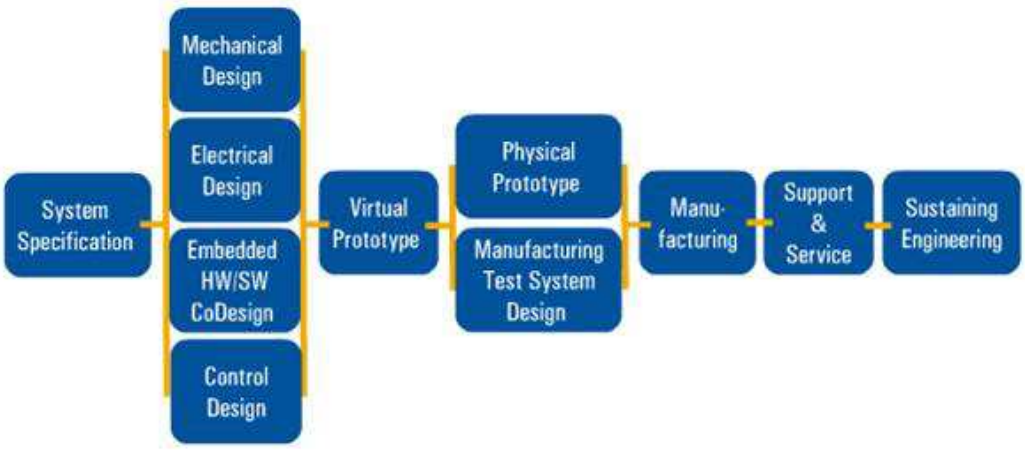


Figura 21. Metodología del diseño Mecatronico, tomado de

http://mecatronicaecuador.com/educacion/file.php/1/MetodologiaMecatronico.JPG

Diseño mecatronico:

- Integración de componentes (hardware).
- Compacto.
- Mecanismos simples.
- Comunicación vía bus o wireless.
- Integración mediante procesos informáticos (software).
- Construcción elástica.
- Retroalimentación programable y control en lazo cerrado.
- Precisión a través de medidas.
- Supervisión con diagnóstico de error.
- Habilidades de aprendizaje.

3.1 MECANICA

La Mecánica, es la ciencia que estudia las leyes generales del movimiento de los cuerpos materiales en relación con las fuerzas que lo producen, estableciendo procedimientos y métodos generales de análisis y de resoluciones de problemas relacionados con esos movimientos.

3.1.1 Materiales mecánicos

TABLA 3. Materiales mecánicos

Material	Cantidad	Precio
Tubo PVC	4 metros	10 dólares
Base PVC	1 unidad	2 dólares
Codo PVC	1 unidad	0.80 dólares
Cable gemelo	5 metros	5 dólares
Cable UTP	5 metros	8 dólares
Tornillos 1cm	100 unidades	3.5 dólares
Tornillos 10cm	4 unidades	0.50 dólares
Base de madera	1 unidad	3 dólares
Aluminio	32 cm	2 dólares
Aluminio	20 cm	1.5 dólares
Tubo de aluminio	1 metro	3 dólares
Platinas de apoyo	3 unidades	3 dólares
Caja de cableado	1 unidad	7 dólares
Refrendado del PVC	3 unidades	45 dólares
TOTAL MATERIALES MECANICOS		94.30 DOLARES

Se decidió construir la parte mecánica a partir de elementos reciclables como es tubo PVC, obtenido de restos de canales para la lluvia de las casas, con el fin de contribuir con la sociedad y el planeta.

Se opto por un diseño tipo SCARA el cual proporciona 3 grados de libertad suficientes para cumplir con el objetivo de un robot manipulador para obtener y clasificar objetos. Además de un diseño atractivo para el usuario.

3.2 ELECTRONICA

La electrónica es el campo de la ingeniería y de la física aplicada relativo al diseño y aplicación de dispositivos, por lo general circuitos electrónicos, cuyo funcionamiento depende del flujo de electrones para la generación, transmisión, recepción, almacenamiento de información, entre otros. Esta información puede consistir en voz o música como en un receptor de radio, en una imagen en una pantalla de televisión, o en números u otros datos en un ordenador o computadora.

3.2.1 Materiales electrónicos

TABLA 4. MATERIALES ELECTRÓNICOS

Material	Cantidad	Precio
Baquelita 20X15 cm	1 unidad	19 dólares
Resistencias 330 ohm	30 unidades	2 dólares
Resistencias 1 Kohm	10 unidades	0.50 dólares
Resistencias 470 ohm	10 unidades	0.50 dólares
Borneras 2 entradas	50 unidades	12.5 dólares
Pic 16f628A	3 unidades	18 dólares
Pic 16f877A	1 unidad	8 dólares

IC 7805	4 unidades	2 dólares
4Mhz cristales	4 unidades	2 dólares
LCD 2 filas	1 unidad	12 dólares
Borneras tipo banana M	10 unidades	3 dólares
Borneras tipo banana F	10 unidades	3 dólares
Conector LDB9	1 unidad	0.70 dólares
Tip 122	10 unidades	3 dólares
Capacitor	5 unidades	1 dólar
Rele 12V	1 unidad	1 dólar
Pulsadores grandes	10 unidades	15 dólares
Caja de circuitos	1 unidad	20 dólares
TOTAL MATERIALES ELECTRICOS		109 dólares

En la electrónica se utilizo placas de control diseñadas en ARES con sus respectivos componentes y sus respectivos valores obtenidos de los cálculos realizados, se deberá utilizar una caja conde irán las tarjetas de control hecha de acrílico para evitar cualquier cortocircuito, además de ello se utilizaron elementos de bajo costo y de alta duración los cuales son perfectos para la realización del prototipo, se tomo en cuenta en la utilización de los microcontroladores que son los encargados de distribuir la información y ejecutar las acciones para el movimiento del prototipo, los cuales son PIC 16 F628A el encargado de transmitir la información de cada servomotor es decir para cada eje un microcontrolador como son 3 son 3 microcontroladores PIC 16f28A, y como cerebro principal el microcontrolador PIC16f628A el cual ejecuta el Menú principal del prototipo y se encarga de relazar las acciones que se transmitan atraves del computador y del sistema de control.

3.2.2 Pic 16f628A

Un microcontrolador es un dispositivo electrónico capaz de llevar acabo procesos lógicos. Estos procesos o acciones son programados en lenguaje ensamblador por el usuario, y son introducidos en este a travez de un procesador. El pic16f628a es un microcontrolador de 8 bit, fue utilizado porque posee una arquitectura RISC avanzada así como un juego reducido de 35 instrucciones. Este microcontrolador es el remplazo del obsoleto pic16f84a, los pines del pic16f628a son compatibles con el pic16f84a, así se podrían actualizar proyectos que he utilizado con el pic16f84a.

Caracteristicas:

- Empaquetado: DIP 18 pines.
- Memoria FLASH : 2K
- Memoria RAM : 224 bytes.
- Memoria EEPROM : 128 bytes.
- Oscilador interno de 4MHz.
- Dos comparadores análogicos.
- USART

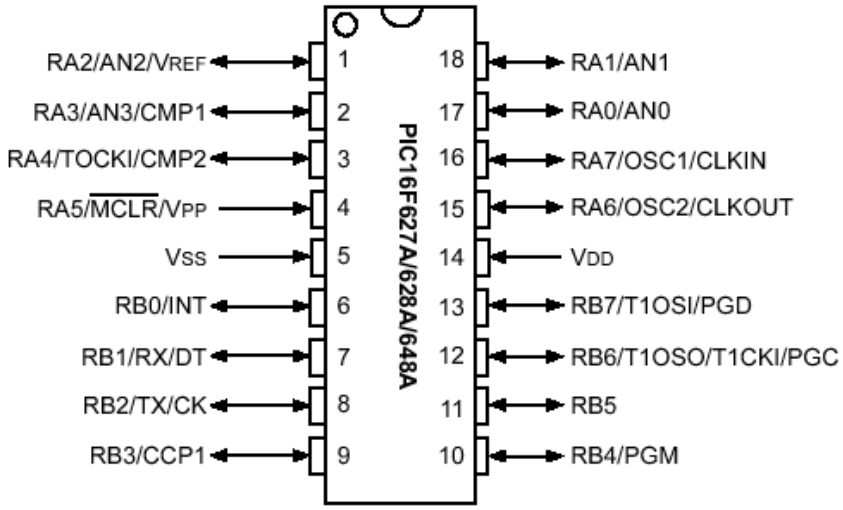


Figura 22. PIC 16f628A numero de pines y puertos, imagen tomada hoja de datos del PIC.

Se utilizo el PIC 16f628A para el control de los servo motores ya que tenemos 3 motores se necesitan 3 PIC, ya que los pic trabajan de modo serial, por este motivo un solo pic no puede tener activado 3 servomotores ya que la salida es PWM y se debe el pulso para poder estar en la posición indicada.

Además utilizamos conexión serial entre el PIC de control que es un PIC 16f877A y los de los servos, ya que así se recibe un dato con la información de la posición de los servomotores.

En el programa PIC BASIC PRO la opción de comunicación serial son las siguientes:

Como se trata de una transmisión serial, siempre va la librería modedefs.bas. No olvidar que la velocidad de envío debe ser la misma que la velocidad de recepción (2400 en este caso).

SERIN portb.0,T2400,valor:

Esta es la instrucción en PIC BASIC PRO que permite recibir un dato mediante comunicación serial, quiere decir que por el puerto B.0 se recibirá

un dato a una velocidad de baudios de 2400, y se guardara el dato en la variable valor. Ejemplo para recibir un dato vía serial:

```
include "modedefs.bas"

valor var byte

trisb =%00000001

portb =%00000000

inicio:

serin portb.0,T2400,valor

if valor == "E" then high portb.1 'E->Encendido

if valor == "A" then low portb.1  'A->Apagado

goto inicio
```

Llamamos a la librería de control y comunicación serial modedefs.bas.

Declaramos la variable valor como byte es decir que puede tomar valores de 0 a 255.

Declaramos que el puerto B tendrá una entrada y 7 salidas, la entrada será PUERTO B.0.

Colocamos el puerto B en estado cero.

Creamos una etiqueta llamada inicio en la cual estará el programa en si

Llamamos a la instrucción SERIN quiere decir que por el puerto B.0 se recibirá un dato a una velocidad de baudios de 2400, y se guardara el dato en la variable valor, mientras no llegue un dato vía serial el programa se quedara en esta instrucción solo si se desconecta o se restea con el máster reset saldrá de la instrucción.

Al llegar el dato comparamos si es E encendido y si es A apagado, el puerto de salida es el PUERT B.1 donde está conectado el LED de aviso.

Y volvemos a la etiqueta inicio a esperar un nuevo dato.

3.2.3 Pic 16f877A

El microcontrolador PIC16F877 de Microchip pertenece a una gran familia de microcontroladores de 8 bits (bus de datos) que tienen las siguientes características generales que los distinguen de otras familias:

- Arquitectura Harvard
- Tecnología RISC
- Tecnología CMOS

En este proyecto se utilizo el PIC 16F877. Este microcontrolador es fabricado por MicroChip familia a la cual se le denomina PIC. El modelo 16F877 posee varias características que hacen a este microcontrolador un dispositivo muy versátil, eficiente y práctico para ser empleado en la aplicación que posteorimente será detallada.

Algunas de estas características se muestran a continuación:

- Soporta modo de comunicación serial, posee dos pines para ello.
- Amplia memoria para datos y programa.
- Memoria reprogramable: La memoria en este PIC es la que se denomina FLASH; este tipo de memoria se puede borrar electrónicamente (esto corresponde a la "F" en el modelo).
- Set de instrucciones reducido (tipo RISC), pero con las instrucciones necesarias para facilitar su manejo.

TABLA 5. Según Descripción General del PIC16F877 en la Tabla de las familias de microcontroladores según sus bits, tomada de DataSheet del PIC.

Subfamilia	Bits del bus de instrucciones	nomenclatura
Base - Line	12	PIC12XXX y PIC14XXX
Mid – Range	14	PIC16XXX
High - End	16	PIC17XXX y PIC18XXX

Variantes principales

Los microcontroladores que produce Microchip cubren un amplio rango de dispositivos cuyas características pueden variar como sigue: - Empaquetado (desde 8 patitas hasta 68 patitas)

- Tecnología de la memoria incluída (EPROM, ROM, Flash)

- Voltajes de operación (desde 2.5 v. Hasta 6v)

- Frecuencia de operación (Hasta 20 Mhz)

Empaquetados Aunque cada empaquetado tiene variantes, especilmente en lo relativo a las dimensiones del espesor del paquete, en general se pueden econtrar paquetes tipo PDIP (Plastic Dual In Line Package), PLCC (Plastic Leaded Chip Carrier) y QFP (Quad Flat Package), los cuales se muestran en las figuras siguientes:

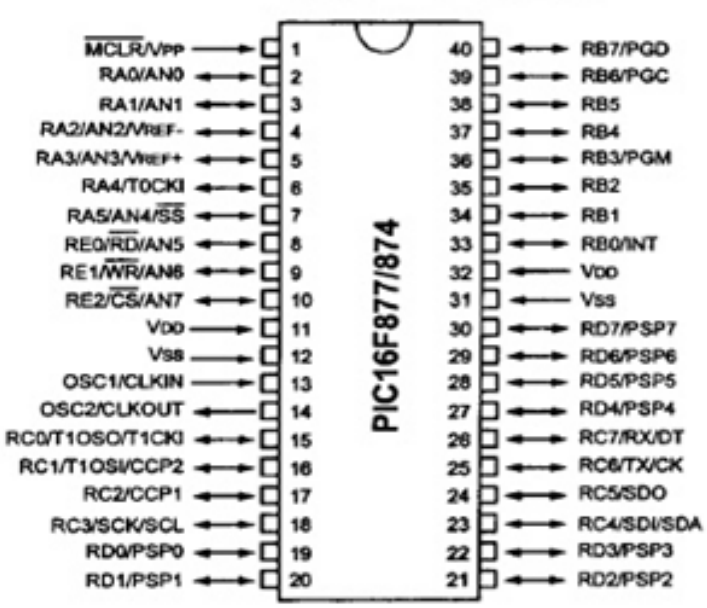


Figura 23. Pic 16f877A encapsulados y tipos de encapsulados, imagen tomada hoja de datos del PIC

Nomenclatura

Además de lo mostrado en la tabla anterior, en el nombre específico del microcontrolador pueden aparecer algunas siglas como se muestra en la siguiente tabla:

TABLA 6. Nomenclaturas

Tipo de memoria	Rango de voltaje	
	Estándar	Extendido
EPROM	PIC16**C**XXX	PIC16**LC**XXX
ROM	PIC16**CR**XXX	PIC16**LCR**XXX
Flash	PIC16**F**XXX	PIC16**LF**XXX

En la siguiente tabla se especifican los rangos de voltaje estándar y extendido manejados por los dispositivos:

TABLA 7. RANGOS DE VOLTAJE

Rango de voltaje	EPROM		ROM		Flash	
Estándar	C	4.5 a 6v	CR	4.5 a 6v	F	4.5 a 6v
Extendido	LC	2.5 a 6v	LCR	2.5 a 6v	LF	2 a 6v

En la siguiente figura se muestra a manera de bloques la organización interna del PIC16F877, Se muestra también junto a este diagrama su diagrama de patitas, para tener una visión conjunta del interior y exterior del Chip.

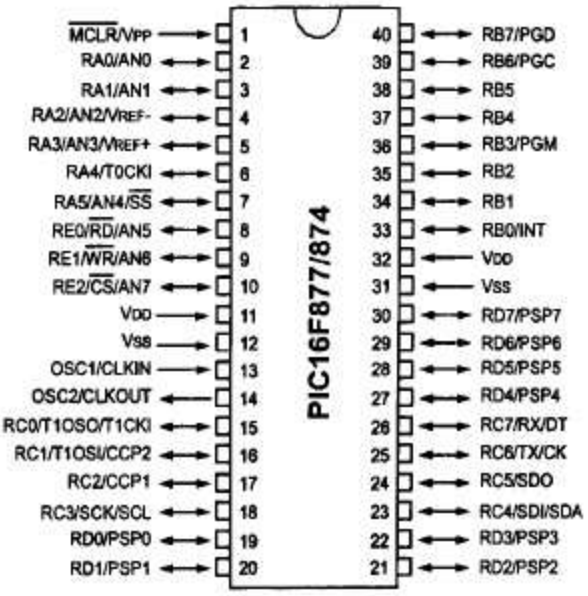


Figura 24. Organización interna del PIC16F877, imagen tomada de hoja de datos del PIC.

Se uso el PIC 16f877A como el control electrónico del brazo tipo scara ya que nos permite utilizar todos sus puertos y es un pic de muy buenas características como las que se nombraron anteriormente, este controlador se conecta directamente con la PC para un funcionamiento controlado del brazo y también controla el brazo de manera autónoma.

Mediante conexión serial con la PC se recibirán las instrucciones que deberá realizar el robot manipulador tipo scara, como la posición, velocidad y sistema de visión artificial.

Se enviara la información hacia los PIC 16f628A por conexión serial de PIC a PIC.

3.2.4 Conector LDB9 y MAX232

Es un adaptador de tensión TTL a niveles adecuados para transmisión serial. Fue utilizado porque al comunicar un microcontrolador con el pic se necesito llevar los 5v del micro a las tensiones con la que trabaja el puerto serial.

Este conector es el que permite la conexión serial entre la PC y la caja de control es decir y el PIC 16f877a, mediante una interfaz de rs232 con el MAX232, estos una conversión TTL a 232.

Circuito integrado para conversión de niveles.

El MAX232 es un circuito integrado que convierte los niveles de las líneas de un puerto serie RS232 a niveles TTL y viceversa. Lo interesante es que sólo necesita una alimentación de 5V, ya que genera internamente algunas tensiones que son necesarias para el estándar RS232. Otros integrados que manejan las líneas RS232 requieren dos voltajes, +12V y -12V.

El MAX232 soluciona la conexión necesaria para lograr comunicación entre el puerto serie de una PC y cualquier otro circuito con funcionamiento en base a señales de nivel TTL/CMOS.

El circuito integrado posee dos conversores de nivel TTL a RS232 y otros dos que, a la inversa, convierten de RS232 a TTL.

Estos conversores son suficientes para manejar las cuatro señales más utilizadas del puerto serie del PC, que son TX, RX, RTS y CTS.

TX es la señal de transmisión de datos, RX es la de recepción, y RTS y CTS se utilizan para establecer el protocolo para el envío y recepción de los datos.

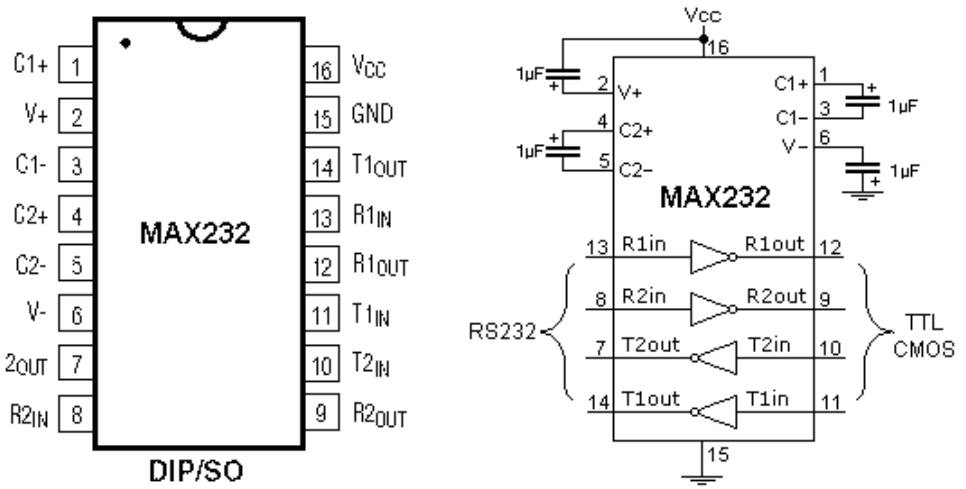
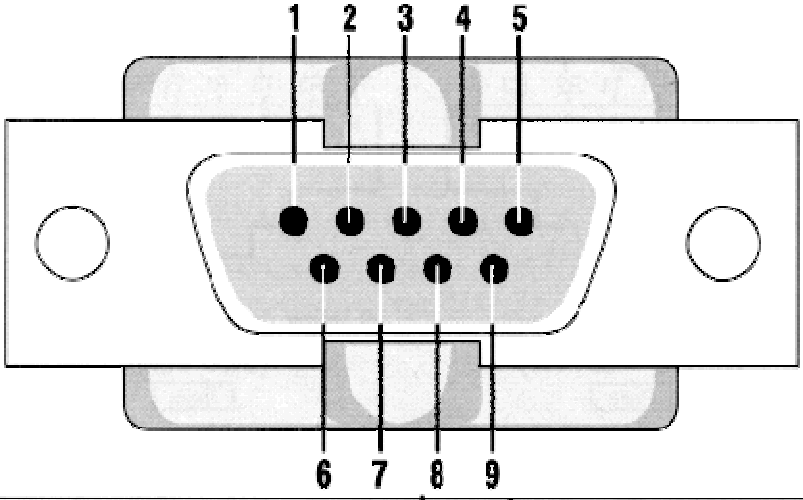


Figura 25. MAX232 configuración interna y de conexión. , imagen tomada http://logica-digital.blogspot.com/2007/11/suplemento-5-las-comunicaciones.html.



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Figura 26. Conector serial y clasificación de sus pines, imagen tomada de http://cybertesis.upc.edu.pe/upc/2007/espinoza_ap/html/sdx/espinoza_ap.html.

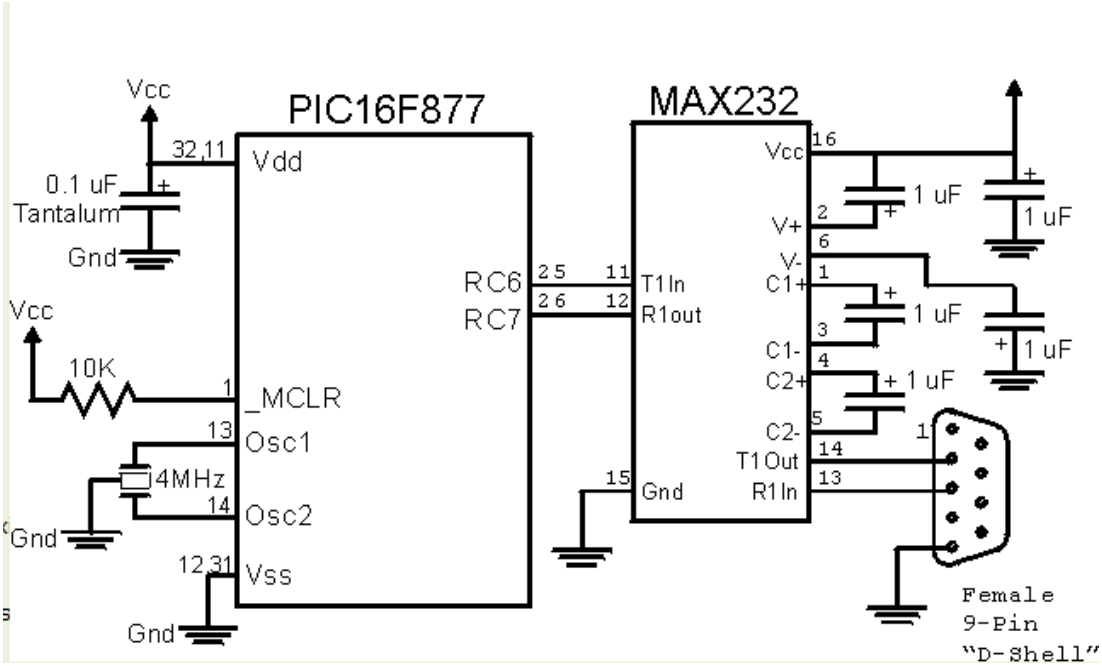


Figura 27. Conexión PIC MAX232 y LDB9, imagen tomada de http://www.ucontrol.com.ar/forosmf/explicaciones-y-consultas-tecnicas/conexion-de-un-max232-a-un-pic16f877/.

3.2.5 PicBasic Pro

El compilador PICBASIC PRO sirve para programar los microcontroladores rapidos y de gran alcance de PICmicro de la tecnología del microchip.

PICBASIC es un programa de Windows. El software especialmente diseñado de editor/interface fue utilizado porque permite corregir el programa con el sintaxis color-coded, salto a las porciones del programa usando un sistema de navegación.

La programación del micro controlador se realizo en la plataforma de MicroCode Studio Pic Basic Pro, ya que es te lenguaje de programación de alto nivel me permite realizar función de manera un veloz y casi exacta, no están preciso en momentos de tiempos como Assembler pero es suficiente

para realizar las operaciones de control y de transferencia de datos por comunicación serial.

El PBP produce código que puede ser programado para una variedad de micro controladores PIC que tengan de 8 a 68 pins y varias opciones en el chip incluyendo convertidores A/D, temporizadores y puertos seriales.

Hay algunos micros PIC que no trabajaran con el PBP, por ejemplo las series PIC 16C5X incluyendo el PIC 16C54 Y PIC 15C58. Estos micro PIC están basados en el viejo núcleo de 12 bit en lugar del núcleo más corriente de 14 bit. El PBP necesita alguna de las opciones que solamente están disponibles con el núcleo de 14 bit como el stack (pila)de 8 niveles.

.Los comandos son enviados al LCD, enviando un $FE seguido por el comando. Algunos comandos útiles se muestran en la siguiente tabla:

TABLA 8. Tabla de LCD en PBP

Comando	**Operación**
$FE, 1	Limpia visor
$FE, 2	Vuelve a inicio primera línea
$FE, $OC	Cursos apagado
$FE, $OE	Subrayado del cursos activo
$FE, $OF	Parpadeo del cursor activo
$FE, $10	Mueve el cursos una posición hacia la izquierda
$FE, $14	Mueve el cursos una posición hacia la derecha
$FE, $C0	Mueve el cursor al principio de la segunda línea

Recibir datos serialmente SERIN:

SERIN Pin,Mode, {Timeout,Label,}{[Qual...],} {Item...}

Recibe uno ó más Items en Pin, en formato standard asincrónico, usando 8 bit de datos, sin paridad y un stop bit (8N1). Include “modedefs.bas” al comienzo del programa PBP. BS1DEFS, BAS y BS2DEFS.BAS ya incluyen MODEDEFS.BAS. Para poder realizar la conexión serial.

A continuación en la **TABLA 9.** se observa el rango y el numero, modo, estado que se utiliza para la emisión de datos, y en la **TABLA 10.** se observa numero, modo, y estado de recepción de datos serialmente.

TABLA 9. TABLA DE DE RECIBIR DATOS SERIALMENTE

	MODE NO	**BAUD RATE**	**STATE**
T2400	0	2400	VERDADERO
T1200	1	1200	VERDADERO
T9600	2	9600	VERDADERO
T300	3	300	VERDADERO
N2400	4	2400	FALSO
N1200	5	1200	FALSO
N9600	6	9600	FALSO
N300	7	300	FALSO

TABLA 10. TABLA DE DE EMITIR DATOS SERIALMENTE

MODE	**MODE NO**	**BAUD RATE**	**STATE**
T2400	0	2400	Llevado a cierto
T1200	1	1200	Llevado a cierto
T9600	2	9600	Llevado a cierto
T300	3	300	Llevado a cierto
N2400	4	2400	Llevado a invertido
N1200	5	1200	Llevado a invertido
N9600	6	9600	Llevado a invertido
N300	7	300	Llevado a invertido
OT2400	8	2400	Abierto cierto
OT1200	9	1200	Abierto cierto
OT9600	10	9600	Abierto cierto
OT300	11	300	Abierto cierto
ON2400	12	2400	Abierto invertido
ON1200	13	1200	Abierto invertido
ON9600	14	9600	Abierto invertido
ON300	15	300	Abierto invertido

3.3 CONTROL

La manipulación indirecta de las magnitudes de un sistema planta (proceso que se desea controlar, manipular sus magnitudes) a través de otro sistema llamado sistema de control(eléctrico, mecánico, neumático).

3.3.1 Materiales de control

Matlab costo de licencia profesional del programa de control es de 6000 dólares.

Se utilizara matlab ya que este sistema de programación es de fácil utilización además tienes mayor capacidad y herramientas en el área de manejo y tratamiento de imágenes, lo que es una ventaja para el desarrollo del prototipo.

El sistema de control se diseño en matlab con el objetivo de realizar 2 funciones esenciales para el funcionamiento del prototipo, la primera se encarga de reconocer palabras la cuales identificaran al depósito de piezas, y envía un dato al cerebro del prototipo es decir al PIC 16f877A y le indica en que posición esta cada uno de los contenedores, la segunda identifica los colores de cada pieza y envía un dato al cerebro para ponerle en cada pieza en su contenedor.

3.4 Cinemática de los manipuladores

Cinemática: Estudio del movimiento sin considerar las fuerzas que lo producen. Propiedades geométricas y temporales. Posición, velocidad, aceleración, derivadas superiores de la posición, etc.

Cinemática de los manipuladores: Propiedades geométricas y temporales del movimiento.

3.4.1 Cinemática directa

Se utiliza fundamentalmente el algebra vectorial y matricial para representar y describir la localización de un objeto en el espacio tridimensional con respecto a un sistema de referencia fijo. Dado que un robot se puede considerar como una cadena cinemática formada por objetos regidos o eslabones unidos entre sí mediante articulaciones, se puede establecer un sistema de referencia fijo situado en la base del robot y describir la localización de cada uno de los eslabones con respecto a dicho sistema de referencia. De esta forma, el problema cinemático directo se reduce a encontrar una matriz homogénea de transformación K que relacione la posición y orientación del extremo del robot respecto del sistema de referencia fijo situado en la base del mismo. Esta matriz K será función de las coordenadas articulares.

En cinemática directa se conocen las variables articulares de una cadena de enlaces de un brazo articulado, su cálculo es sencillo (multiplicación matricial)

Para calcular la posición X, Y del brazo utilizando método geométrico se tienen las siguientes ecuaciones:

$$x1 = d1 \times sen(q1) + d2 \times sen(q1 + q2)$$ **Ecuación** [3.1]

$$y1 = d1 \times cos(q1) + d2 \times cos(q1 + q2)$$ **Ecuación** [3.2]

Con estas ecuaciones graficamos el espacio de trabajo en el plano (X-Y) utilizando MATLAB.

Donde:

d1: eslabón del brazo uno o mayor

d2: eslabón del antebrazo uno o menor

q1: ángulo uno

q2: ángulo dos

El cálculo de la posición en Z del robot scara se define considerando la posición mínima d3 desde el origen del robot y el desplazamiento q3 dado por el efector fina.

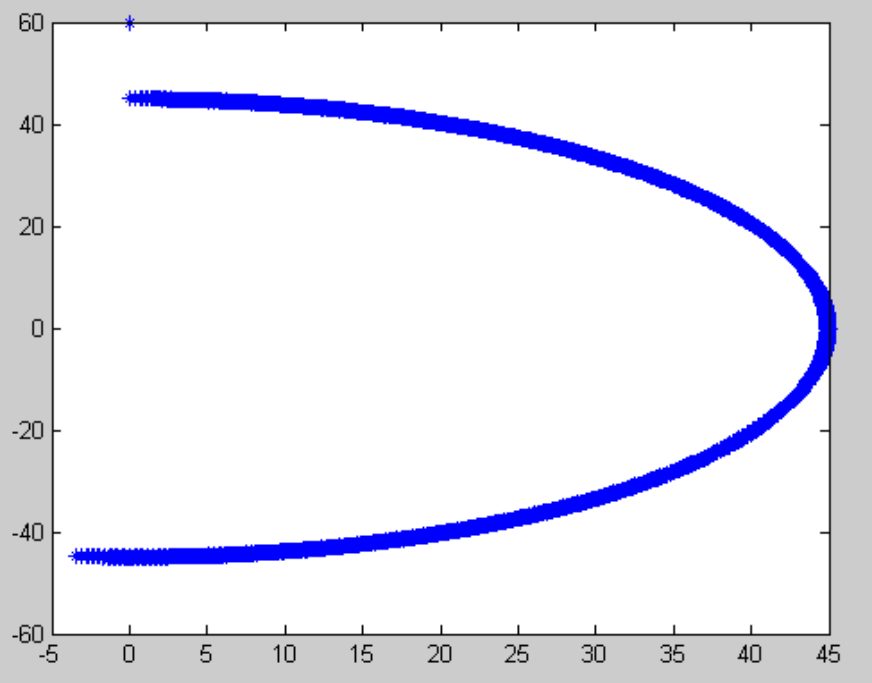


Figura 1. Espacio de trabajo del robot tipo escara simulado en matlab.

3.4.2 Cinemática Inversa

El objetivo de la cinemática inversa consiste en encontrar la posición que deben adoptar las diferentes articulaciones dado solamente la posición del efector final.

El método permite transformar un sistema de coordenadas de un vector referido a un sistema de coordenadas homogéneas de forma que:

$$T = \begin{Bmatrix} R_{3X3} & P_{3X1} \\ f_{1X3} & w_{1X1} \end{Bmatrix}$$ **Ecuación** [3.3]

Donde:

- R indica un vector de rotación
- P indica un vector de traslación
- f indica un vector de Perspectiva
- W indica un vector Escalado

De esta manera se puede establecer una matriz homogénea que contenga la información para cada uno de los sistemas de coordenadas:

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ **Ecuación**[3.4]

Las ecuaciones dadas a MATLAB fueron:

Las ecuaciones para saber el área de trabajo del robot scara.

x1=(d1*sind(q1)+d2*sind(q1+q2)) **Ecuación** [3.5]

y1=(d1*cosd(q1)+d2*cosd(q1+q2)) **Ecuación** [3.6]

Teniendo el área de trabajo de obtenido atraves de la una imagen, comenzamos a trazar el centro de cada pieza inmersa en el área de trabajo para ello utilizamos la función de centroide de matlab, y encontramos los puntos de cada centroide X y Y desde el punto cero de la imagen.

Trazar coordenadas del punto X, del punto Y y la hipotenusa que forma esa coordenada.

Obtener el ángulo alfa y beta para obtener la posición del brazo y calcular el ángulo delta para obtener la posición del antebrazo.

Teniendo en cuenta las medidas del área de donde va a trabajar el brazo, para ello hay que transformar los pixeles en cm, es decir la imgen trabaja con pixeles(n m) y en la realidad tendemos que trabajar con cm (x y) para ello se toma como referencia el punto cero de la imagen y el punto cero del área de trabajo.

Obteniendo la imagen atraves de una webcam calibrada en el area de trabajo procedemos a utilizar el siguiente código.

“
```
%transformacion a medidas los pixeles

medidaIG=[size(imagengris)]% medida de la imagen gris
 medidaIGY=medidaIG(1)%medida de la y de los pixeles de la imagen gris
 medidaIGX=medidaIG(2)%medida de la x de los pixeles de la imagen gris
 medidaTTY=27;%medida de  la tabla de trabajo y
 medidaTTX=24%mediad de la tabla de trabajo x
 for n=1 :Ne
 dxrR(n)=((dxr(n)*medidaTTX)/ medidaIGX); %calculamos los pixeles por
centimetros de la tabla de trabajo x
 dyrR(n)=((dyr(n)*medidaTTY)/ medidaIGY);%calculamos los pixeles por
centimetros de la tabla de trabajo y
```
”
```
end
```

4. Diseño del Robot Manipulador con Visión Artificial

El diseño de un prototipo de robot manipulador con visión artficial se compone de 3 principales partes la parte mecánica, electrónica y de control.

El robot se encargara de moverse cinemáticamente alrededor de una tablero el cual es su área de trabajo donde se encontraran las piezas que serán clasificadas, atraves del diseño mecánico el cual permite un movimiento de 3 grados de libertad para alcanzar cualquier punto del área de trabajo, con una tarjeta electrónica diseñada para trabajar independientemente de un computador atraves de pulsadores y palancas análogas, también se lo puede conectar a un PC el cual controlara el prototipo.

El sistema en modo controlado, trabaja bajo órdenes del ordenador que es el encargado de enviar las señales ya que este es la visión artificial programada en matlab, el cual realiza las operaciones de leer y reconocer colores.

4.1 Diseño Mecánico

En el diseño mecánico utilizaremos el software SOLID WORKS el cual nos permite diseñar mecanismos en CAD y 3DCAD, simularlos y comprobar su funcionamiento. Además nos permiten diseñar el grafico y los planos para desarrollar el mecanismo en el torno o en la fresa.

SolidWorks es un programa de diseño asistido por computadora para modelado mecánico desarrollado en la actualidad por SolidWorks Corp., una subsidiaria de Dassault Systèmes (Suresnes, Francia), para el sistema operativo Microsoft Windows.

El programa permite modelar piezas y conjuntos y extraer de ellos tanto planos como otro tipo de información necesaria para la producción. Es un programa que funciona con base en las nuevas técnicas de modelado con sistemas CAD. El proceso consiste en trasvasar la idea mental del diseñador al sistema CAD, “construyendo virtualmente” la pieza o conjunto.

Posteriormente todas las extracciones (planos y ficheros de intercambio) se realizan de manera bastante automatizada.

Piezas modeladas en SolidWorks

Se diseño una base circular por el hecho de que la materia prima era tubo PVC, a la cual se la refrendo en un torno para que se encuentre con un estilo uniforme, y poder utilizarle.

En el diseño de la base principal se le ha puesto directamente la simulación del servomotor, para en el momento de realizar el ensamble de las piezas no exista el inconveniente de la falta del espacio o exceso de espacio del servomotor, además de esto sirve para la simulación del diseño.

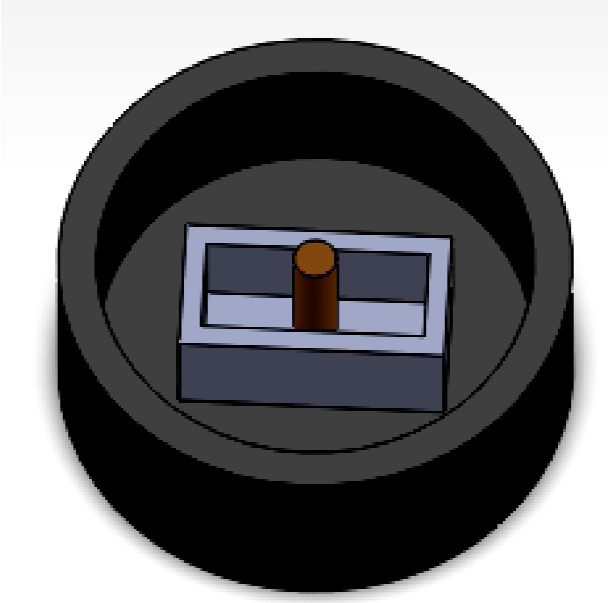


Figura 28. Base diseñada en SolidWorks

Las medidas de la base diseñada se describirán en la siguiente figura:

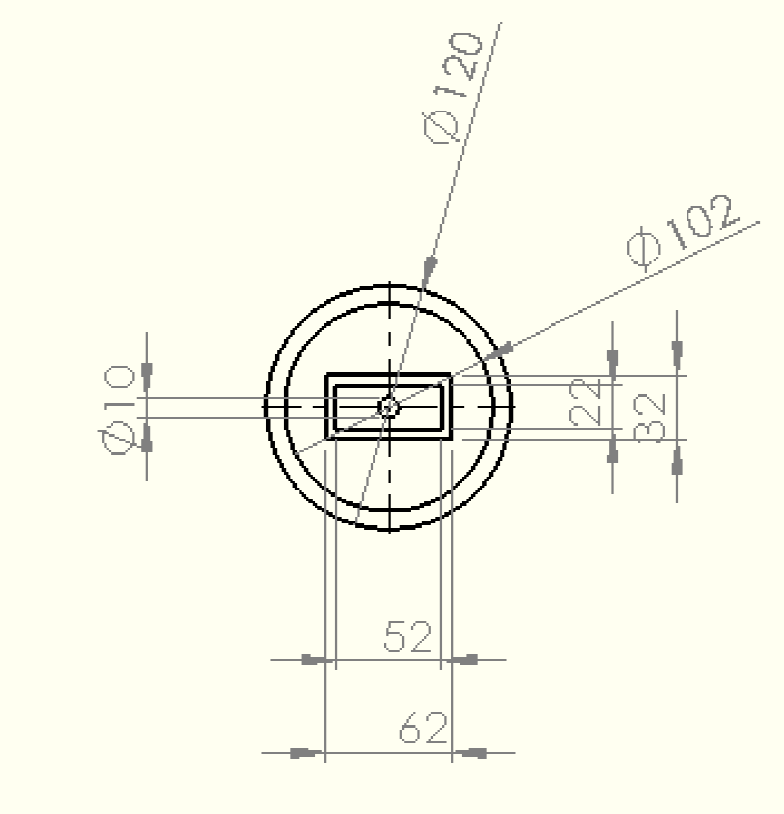


Figura 29. Base principal vista superior

En la figura anterior podemos observar las medidas de la base principal en una vista superior.

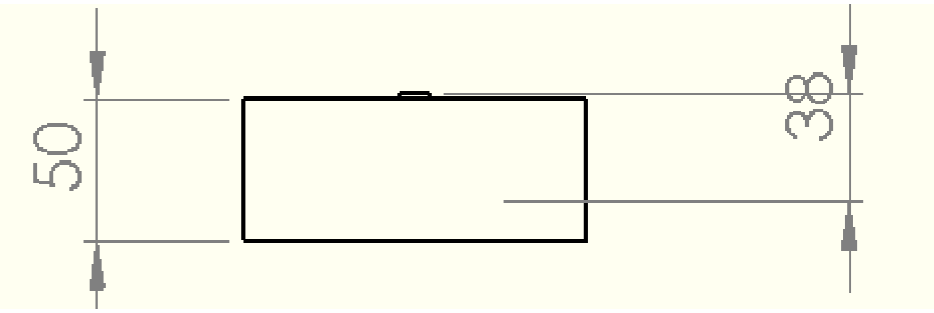


Figura 30. Base principal vista posterior

En la figura anterior podemos ver las medidas que nos faltaba la altura del eje y altura de la base.

Análisis de Esfuerzos para la base principal

La base del robot SCARA soporta una fuerza de 50 N equivalente al peso total del manipulador sin tomar en cuenta el peso de la carga que mueve este manipulador.

El peso (masa*gravedad) de 50N es con el que se han realizado los análisis tanto de esfuerzos como de desplazamiento estático mostrados en las **Figuras 30.1** y **30.2**

Fap. (Fuerza aplicada a la pieza del robot**).**

Fmin. (Carga admisible)**.**

FS. (Factor de seguridad**).**

Tomando en cuenta las distintas fuerzas se realizara el análisis de esfuerzos en el programa CAD solidworks.

FS= Fap. (Carga aplicada a la pieza del robot**) / Fmin. (**Carga admisible)**.**

FS = (50N)/(48N)

FS= (1.04) factor de seguridad de la base. **Ecuación** [4.1]

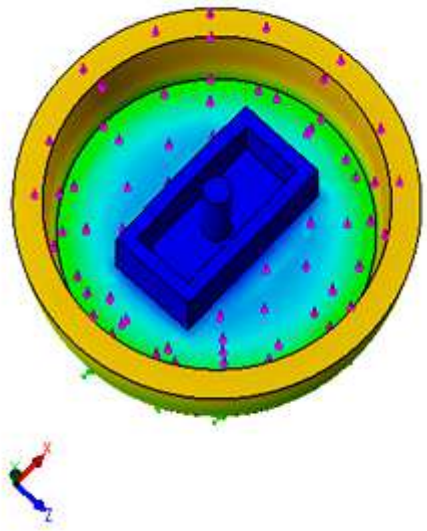


Figura 30.1. Base principal análisis de esfuerzos

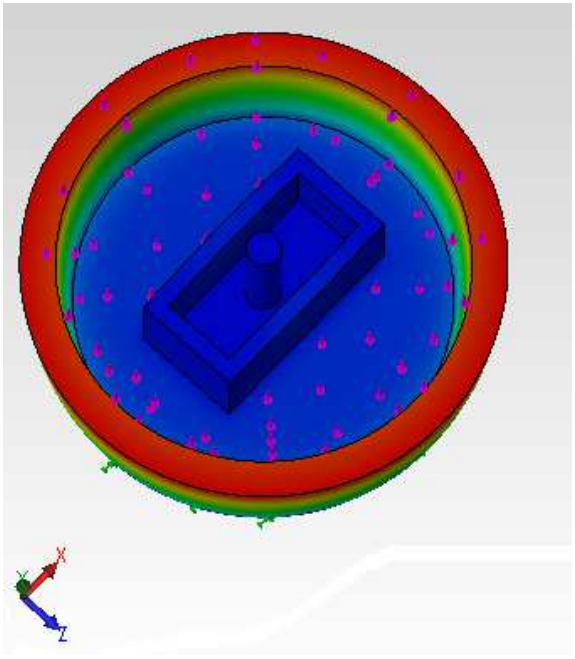


Figura 30.2. Base principal análisis de desplazamiento estático.

Base secundaria

En esta base diseñada igualmente con forma parecida aun acople de tubo PVC, el cual se acopla a la base principal y proporciona la estabilidad del brazo robot, ya que a través de esta pasa el primer eje que moverá el brazo robot. Esta pieza es sumamente necesaria y debe estar completamente bien diseñada por ser la principal en soportar el peso del robot y su carga.

Para soportar el peso aplicado la base secundaria debe estar diseñada de manera que se ajuste lo suficiente a la base primaria y brinde el apoyo al eje primario, para poder soportar los esfuerzos.

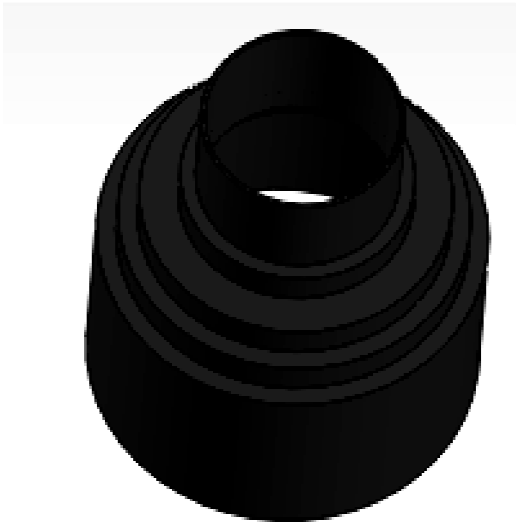
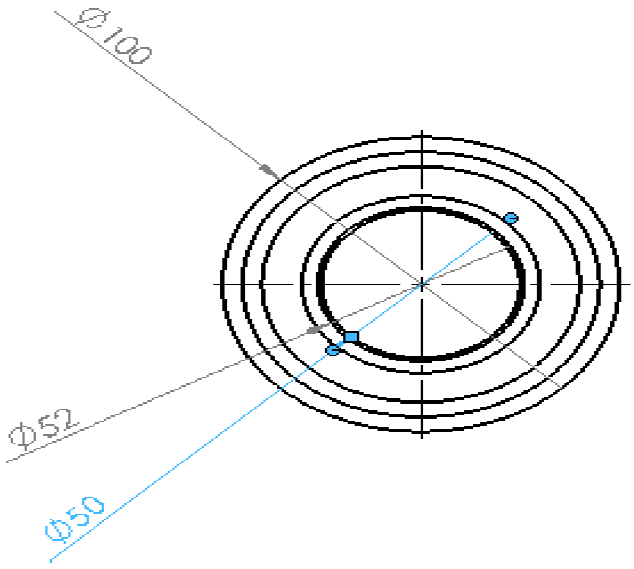


Figura 31. Base superior diseñada en SolidWorks

Las medidas de la base secundaria, diseñada se describirán en la siguiente imagen:



(a)

Figura 32. Vista superior de la Base secundaria

En la figura anterior podemos ver las medidas que se dé la base secundaria como son:

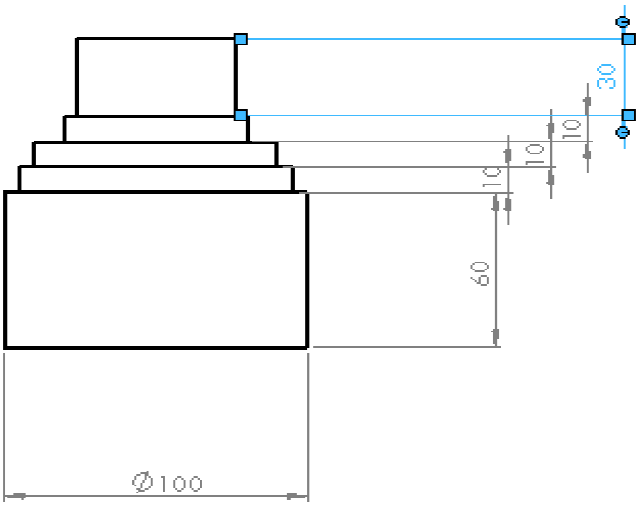


Figura 33. Vista Frontal de la Base secundaria

Análisis de Esfuerzos para la base secundaria

La base secundaria del robot SCARA soporta una fuerza de 50 N equivalente al peso total del manipulador sin tomar en cuenta el peso de la carga que mueve este manipulador.

El peso (masa*gravedad), es con la que se han realizado los análisis tanto de esfuerzos como de desplazamiento estático.

En esta base mantenemos los valores de la base principal ya que igualmente soporta el peso de todo el mecanismo.

La base secundaria hace de guía para eje uno que es el que guía el primer brazo.

Fap. (Fuerza aplicada a la pieza del robot**).**

Fmin. (Carga admisible)**.**

FS. (Factor de seguridad**).**

Tomando en cuenta las distintas fuerzas se realizara el análisis de esfuerzos en el programa CAD solidworks.

FS= Fap. (Carga aplicada a la pieza del robot**) / Fmin. (**Carga admisible)**.**

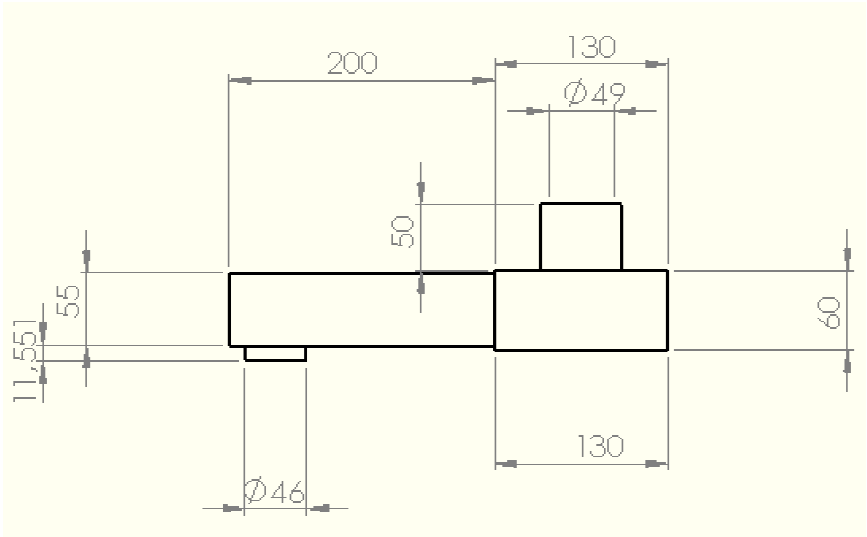
FS = (50N)/(48N)

FS= (1.04) factor de seguridad de la base. **Ecuación** [4.2]

Brazo uno o eslabón uno

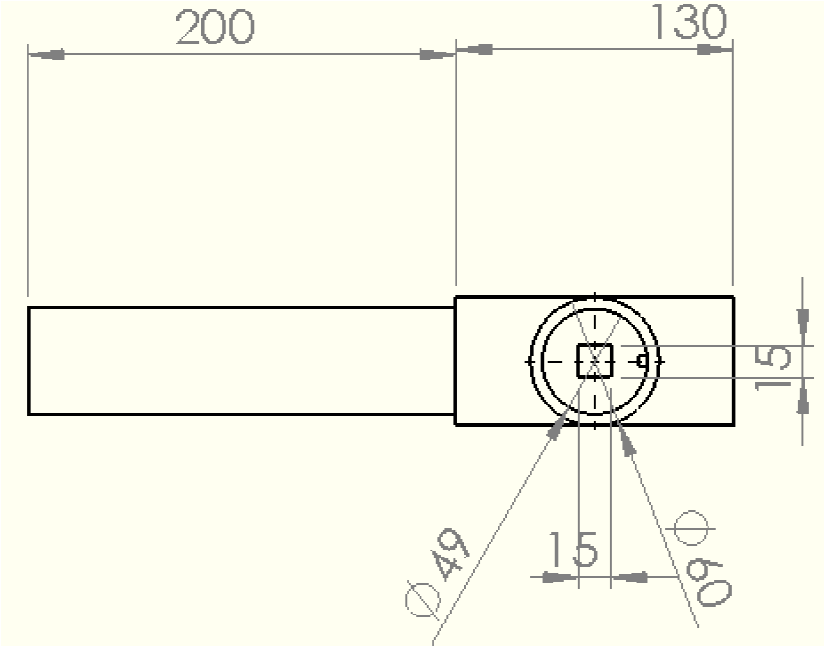
Este brazo es la primera parte de movimiento del robot tipo scara, fue diseñado con una entrada y barios orificios para unirlo al eje principal, que se encuentra en conexión con la base principal y la secundaria, además este brazo es el que lleva el segundo servomotor y el segundo eje en el acople de eje en la parte final del mismo. En esta parte final se encuentra un eje que simula el movimiento del segundo servomotor.

Figura 34. Brazo o eslabón uno con agarre para motor 2



(a)

Figura 35. Vista frontal del Brazo uno



(b)

Figura 36. Vista superior del Brazo uno

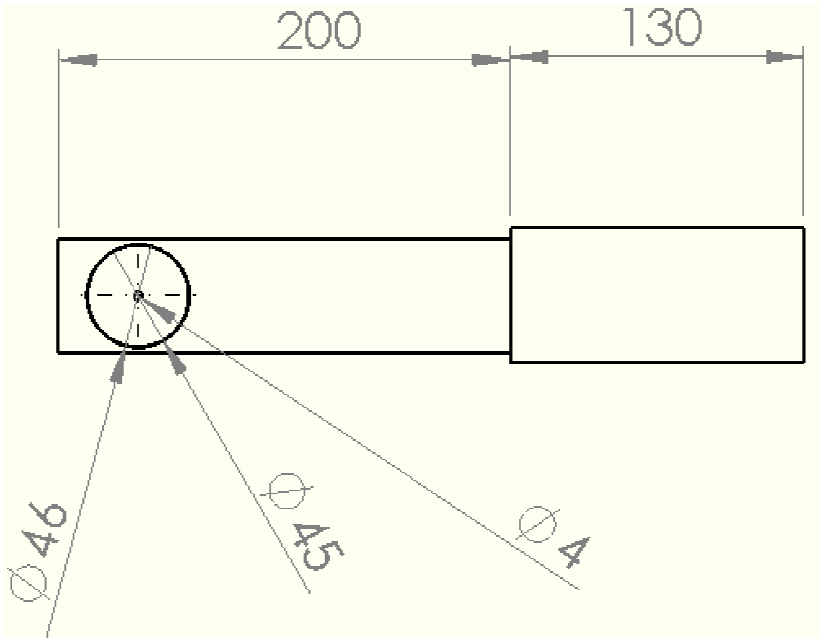


Figura 37. Vista inferior del Brazo uno

Análisis de Esfuerzos para el brazo uno

El primer brazo del robot SCARA soporta una fuerza de 20 N equivalente al peso total del eje secundario, del segundo brazo y del efector final, sin tomar en cuenta el peso de la carga que mueve este manipulador.

Peso (masa*gravedad), y con la que se han realizado los análisis tanto de esfuerzos como de desplazamiento estático.

Fap. (Fuerza aplicada a la pieza del robot**).**

Fmin. (Carga admisible)**.**

FS. (Factor de seguridad**).**

Tomando en cuenta las distintas fuerzas se realizara el análisis de esfuerzos en el programa CAD solidworks.

FS= Fap. (Carga aplicada a la pieza del robot**) / Fmin. (**Carga admisible)**.**

FS = (20N)/(18N)

FS= (1.1) factor de seguridad del brazo. **Ecuación** [4.3]

Eje uno o principal

Este eje de tubo PVC el mismo que se encarga de transmitir el movimiento del servomotor al brazo principal, está diseñado como un tubo para poder refrendarlo en un torno y que la materia prima que es tuvo PVC funcione como un eje diseñado de una pieza entera.

En su base se encuentra un fondo con la conexión al piñón del servomotor es decir se conecta directamente con este, con un ajuste de diseño de fabrica del servomotor, y a sus lados se encuentran agujeros para conectarse con el brazo principal y así transmitir en movimiento en el robot.

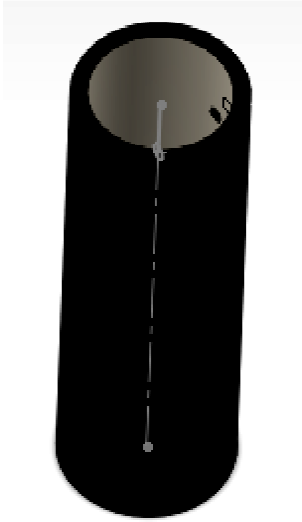


Figura 38. Eje del Brazo o eslabón uno con agarre para motor 2

Las medidas de la base secundaria, diseñada se describirán en la siguiente imagen:

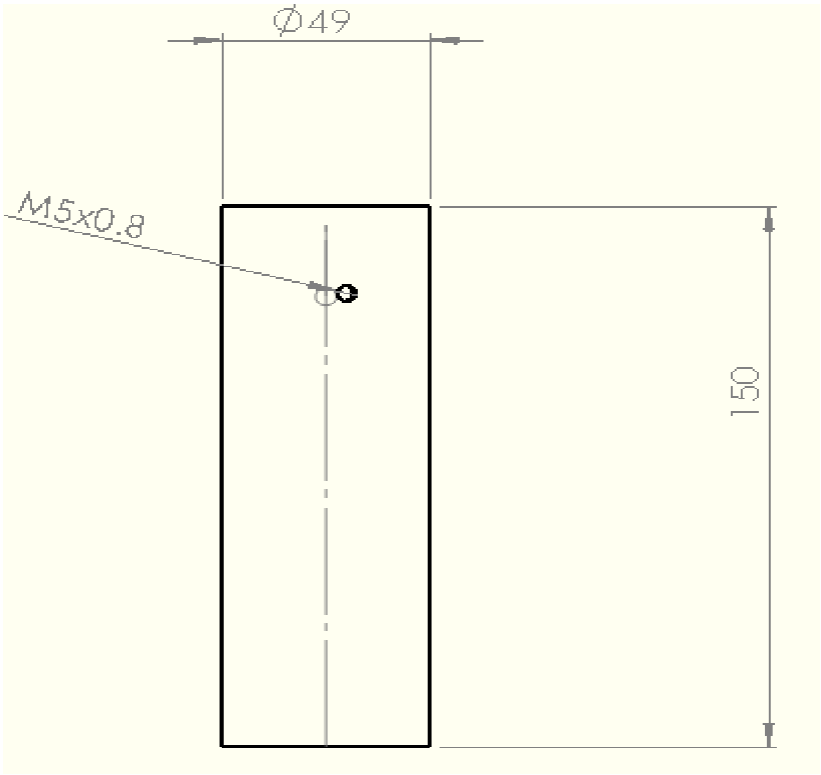
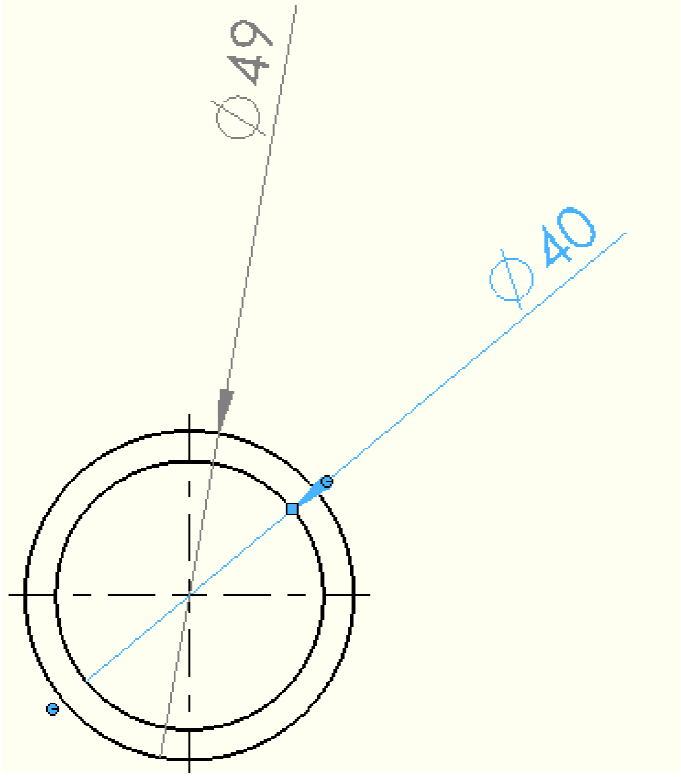
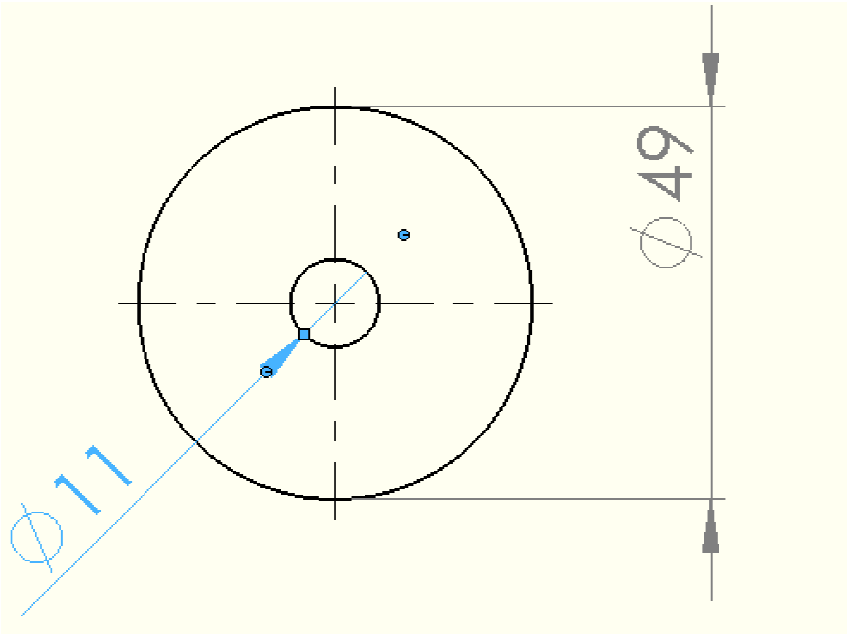


Figura 39. Vista derecha del eje uno



(a)

Figura 40. Vista superior del eje uno



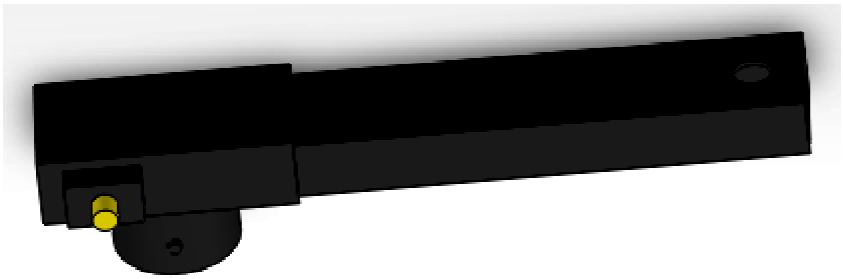
(b)

Figura 41. Vista inferior del eje uno

Brazo secundario

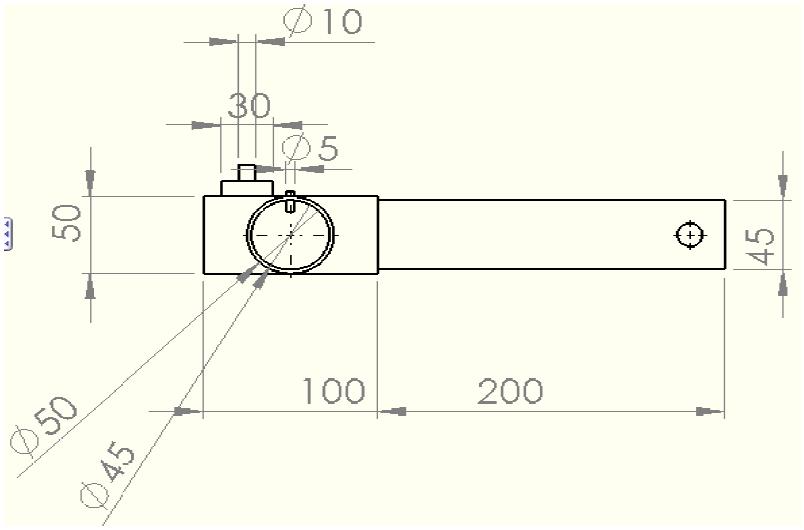
Este diseño se lo realizo con el acople del eje secundario, y la entrada del tercer servomotor que proporciona el movimiento del efector fina, el cual se encuentra en la parte final del brazo secundario, ya que en esta parte se utiliza como guía para subir abajar el efector final.

En el diseño de este brazo igual que en la base principal se lo a incluido al tercer servomotor para simular el movimiento del efector final.



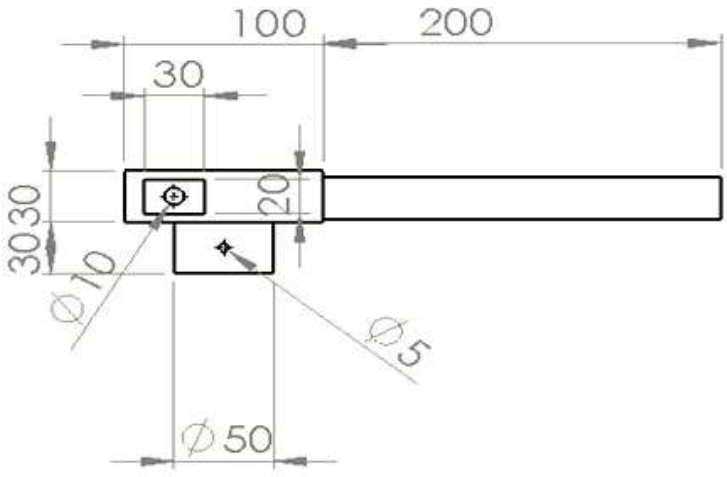
(a)

Figura 42. Brazo o eslabón dos con agarre para actuador y motor de actuador.



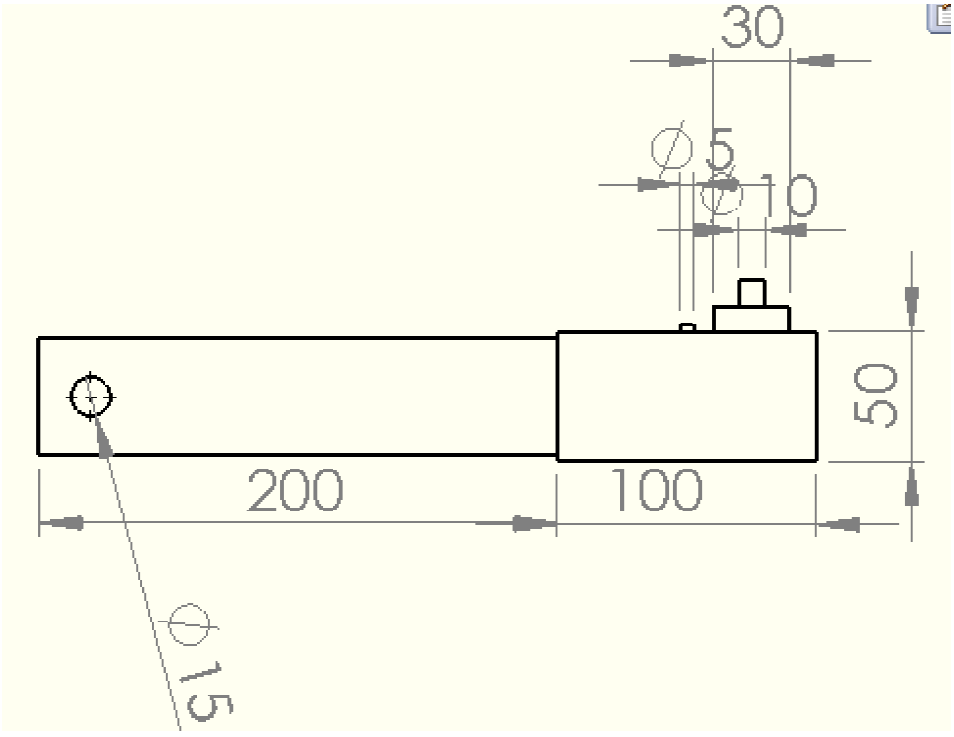
(b)

Figura 43. Vista inferior del Brazo o eslabón dos con agarre para actuador y motor de actuador.



(a)

Figura 44. Vista lateral derecha del Brazo o eslabón dos con agarre para actuador y motor de actuador.



(b)

Figura 45. Vista superior del Brazo o eslabón dos con agarre para actuador y motor de actuador.

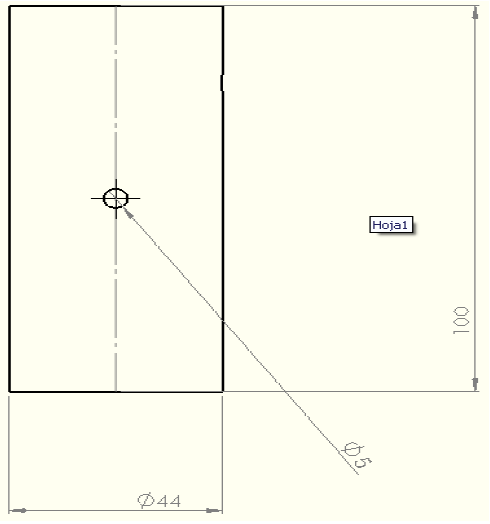
Eje secundario

El eje secundario es el que se encuentra unido al segundo servomotor a través del piñón del servomotor y un engrane de fabrica, este eje está ubicado en la parte final o en la punta de nuestra brazo principal.

La función principal de este eje es transmitir el movimiento al segundo brazo y así obtener nuestro robot tipo scara.

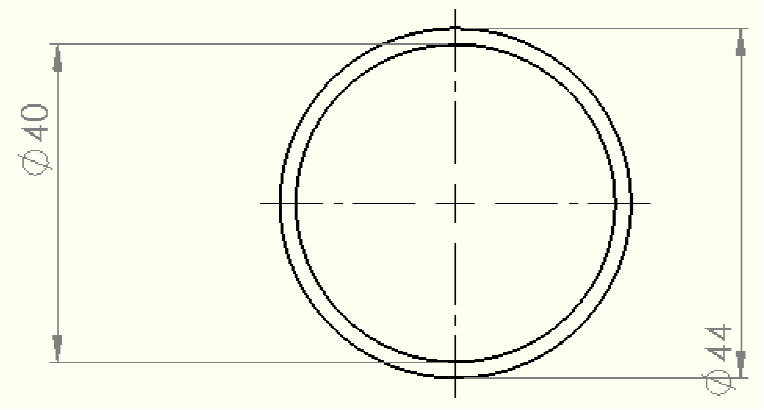
(a)

Figura 46. Eje del Brazo o eslabón dos



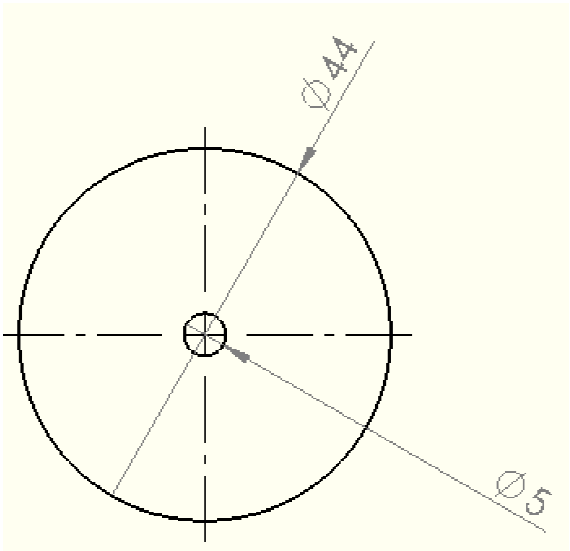
(b)

Figura 47. Vista lateral derecha del eje 2



(a)

Figura 48. Vista superior del eje 2



(b)

Figura 49. Vista inferior del eje 2

Efector Final

El efector final está diseñado como un tubo hueco por el cual pasan 4 cables los cuales son la señal del sensor de llegada, y de un electroimán el cual es el encargado de tomar las piezas.

Además el mecanismo que se encarga de levantarlo son 2 eslabones el de mayor longitud conectado a servomotor del segundo brazo y segundo eslabón conectado al tubo hueco del efector final, este sistema de eslabones funciona como péndulo invertido el cual permite subir y bajar el electroimán.

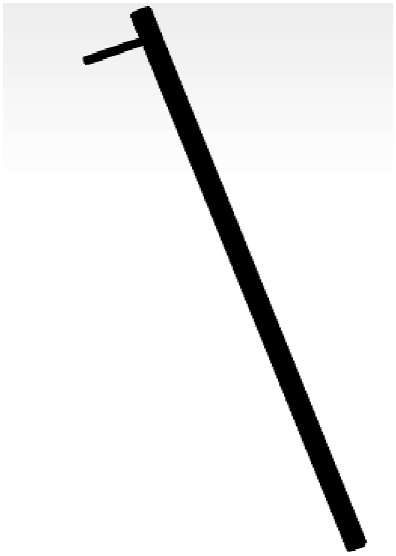
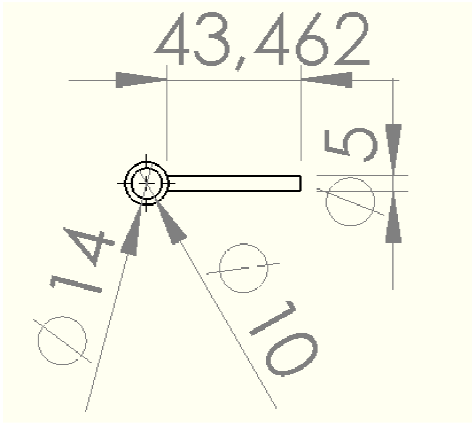


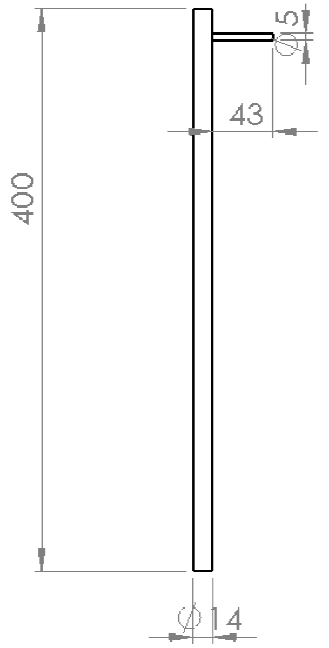
Figura 50. Efector final.

A continuación se mostraran las medidas que se obtienen en el diseño del efector final.



(b)

Figura 51. Vista superior del Efector final.



(c)

Figura 52. Vista lateral del Efector final.

Eslabón uno del mecanismo del Efector Final

Este eslabón es el que permite transmitir el movimiento de subir y bajar el tubo hueco, ya que se encuentra conectado al servomotor en el lado derecho del segundo brazo del robot, y conectado al segundo eslabón que transmite el movimiento en forma de péndulo al tubo hueco.

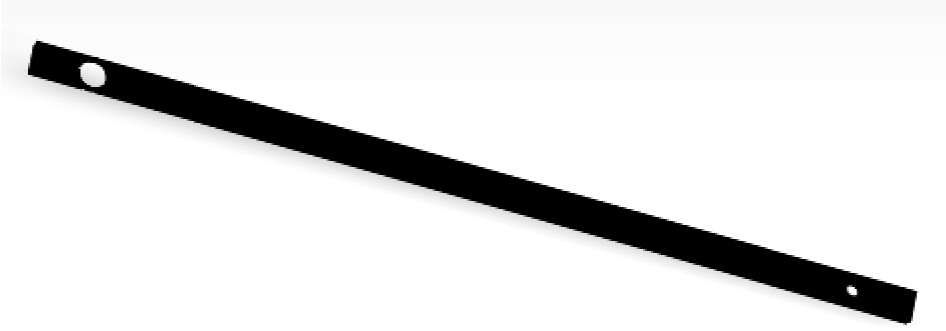


Figura 53. Eslabón uno del efector final

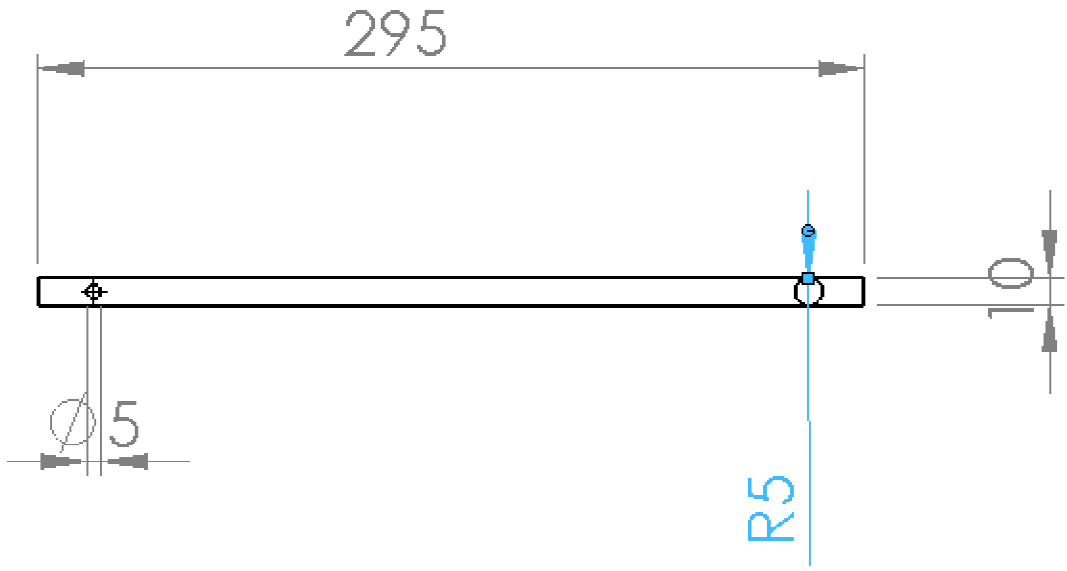


Figura 54. Vista lateral del Eslabón uno del efector final

Eslabón dos del mecanismo del Efector Final

Este eslabón al igual que el anterior transmite el movimiento del servomotor para accionar el movimiento del tubo vacio de subir y bajar, para poder alcanzar las piezas, se conecta directamente con el tubo vacio y con el eslabón numero uno.

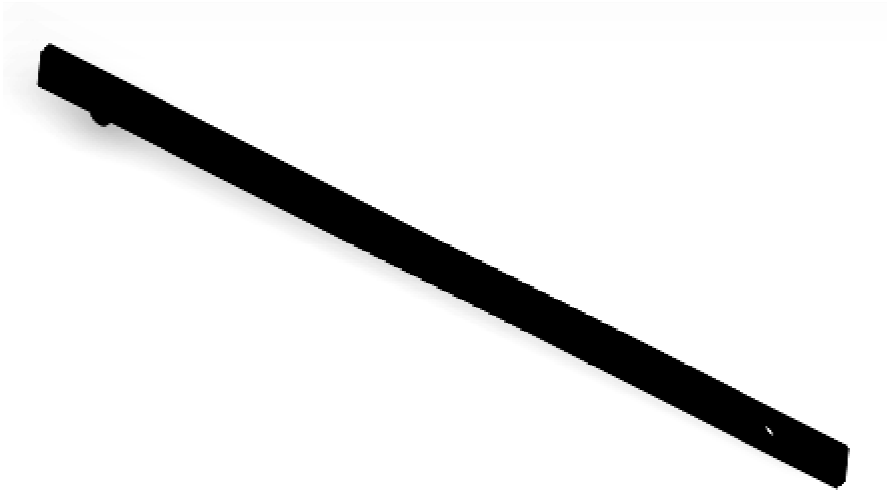


Figura 55. Eslabón dos del efector final

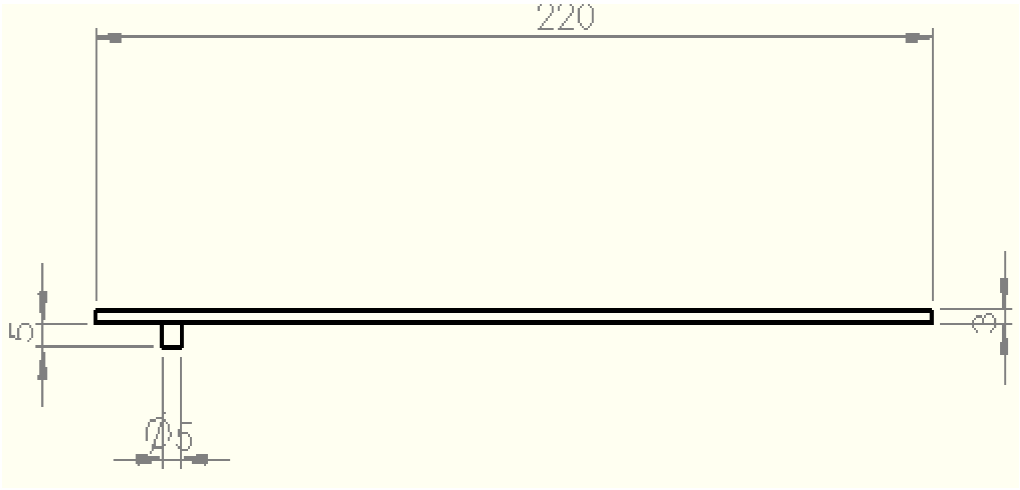


Figura 56. Vista lateral del Eslabón dos del efector final

4.2 Prototipo Virtual

Este mecanismo es la unión de los 3 elementos es decir del eslabón uno, eslabón dos, y tubo vacio.

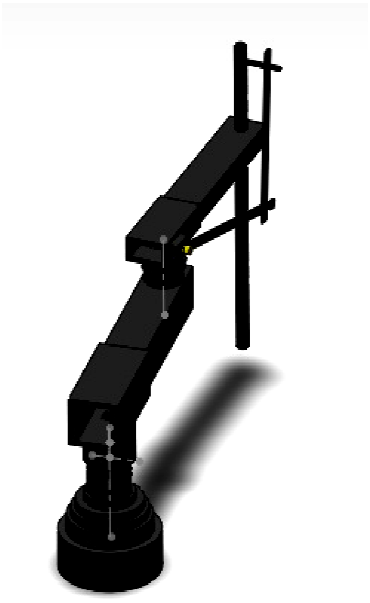


Figura 57. Ensamblaje Brazo tipo scara con efector final vistas.

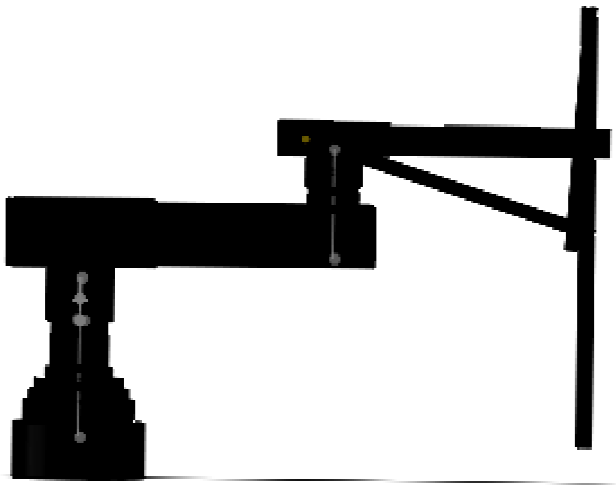


Figura 58. Ensamblaje Brazo tipo scara con efector final vistas.

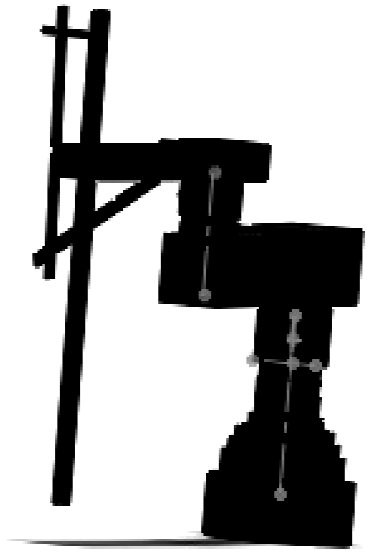


Figura 59. Ensamblaje Brazo tipo scara con efector final vistas.

Validación mecánica del diseño

Mediante al análisis de elemento finito se comprueba el diseño de cada una de las placas que integran los eslabones del robot. Dicho análisis se llevo a cabo en SOLIDWORKS, el cual cuenta con un módulo de análisis de elemento finito. Recordando la distribución de cargas se divide el análisis en cada eslabón del robot.

La distribución de cargas se en lista a continuación:

- Carga máxima a levantar (g L)= (0.500 Kg) (9.81 m/s2) = 4.905 N
- Mecanismo del gripper (g MG)= (1.0 Kg) (9.81 m/s2) = 9.81 N
- Peso del eslabón 2(g E2)= (0.750 Kg) (9.81 m/s2) = 7.36 N
- Mecanismo de eslabón 2(g M3)= (1.0 Kg) (9.81 m/s2) = 9.81 N
- Motor para gripper (g MOT3)= (0.750 Kg) (9.81 m/s2) = 7.36N
- Rodamientos entre eslabón 2 y 1(g ROD)= (0.200 Kg) (9.81 m/s2) = 1.962N
- Peso del eslabón 1(g E1)= (0.750 Kg) (9.81 m/s2) = 7.36 N
- Mecanismo de eslabón 1 (g M1)= (1.0Kg) (9.81 m/s2) = 9.81 N
- Motor para eslabón 2 (g MOT2)= (0.750Kg) (9.81 m/s2) = 7.36 N

Ecuaciónes [4.4]

Análisis para el eslabón 2

Las cargas a las que está sometido este eslabón son: gL, gMG, gE2, gM3, gMOT3. Se sabe que la resistencia del aluminio a la fluencia (σy) es de255 MPa(mega pascales), mismo que es el esfuerzo de falla, y considerando un coeficiente de seguridad de 2, se obtiene lo siguiente:

$$coeficiente\ de\ seguridad = \frac{\sigma_y}{\sigma_{Trabajo}}$$

$$\sigma_{Trabajo} = \frac{\sigma_y}{coeficiente\ de\ seguridad}$$

$$\sigma_{Trabajo} = \frac{255}{2} = 125MPa$$

Ecuación [4.5]

Haciendo un análisis de esfuerzos en SOLIDWORKS, se obtiene los resultados mostrados en la figura 1 del anexo B, donde se puede observar que el esfuerzo máximo al que está sometida la pieza es de 15.43 MPa, esfuerzo mucho menor al esfuerzo de trabajo, por lo que sea segura la durabilidad de éste.

Análisis para el eslabón 1

Las cargas a las que está sometido este eslabón son: gROD, gE1, gM1, gMOT2

Debido a lasfuerzas de reacción que tiene el eslabón 2 sobre el eslabón 1 existe una carga de 39.245 N. Se sabe que la resistencia del aluminio a la fluencia (σy) es de 255 MPa, mismo que es el esfuerzo de falla, y considerando un coeficiente de seguridad de 2, se obtiene lo siguiente:

$$\sigma_{Trabajo} = \frac{255}{2} = 125MPa$$

Ecuación [4.6]

Haciendo un análisis de esfuerzos en SOLIDWORKS, se obtiene los resultados, donde se puede observar que el esfuerzo máximo al que está sometida la pieza es de 53.75 MPa, esfuerzo mucho menor al esfuerzo de trabajo, por lo que se asegura la durabilidad de éste, además la deformación máxima sufrida es de 2.282 mm

4.2.1 Parte electromecánica.

La parte electromecánica del sistema funciona atreves de 3 servomotores de 15Kg por centímetro, los cuales son controlados por ancho de pulso enviados atreves de las tarjetas electrónicas que se definirán más adelante.

Estos servomotores son los que permiten que el sistema pueda moverse de manera controlada sobre el área de trabajo.

En la **Figura 60** se muestra un diseño que realice en solidworks del servomotor utilizado**.**

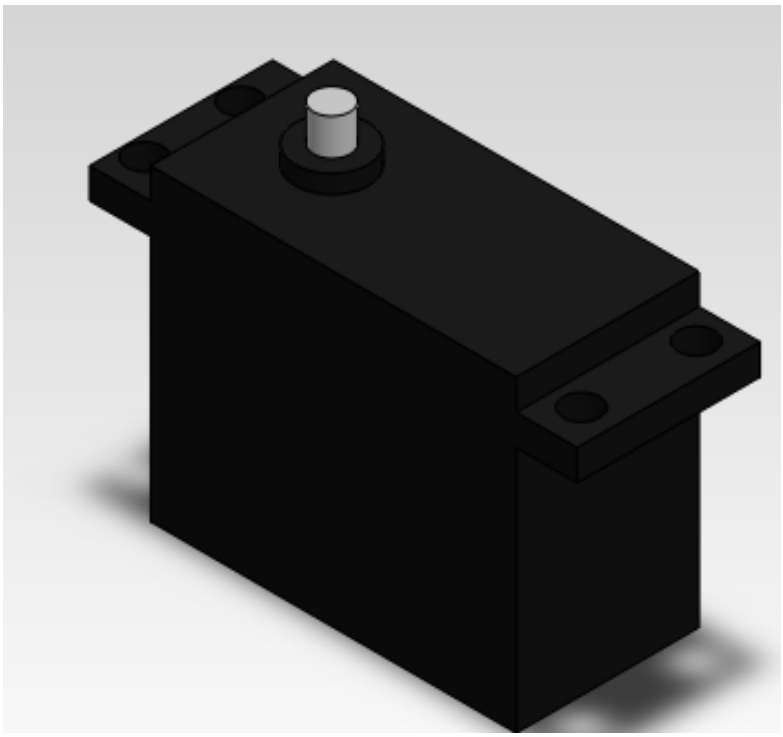


Figura 60. Servomotor

4.2.2 OCR reconocimiento óptico de caracteres

El reconocimiento óptico de caracteres es tomar una imagen en la cual se encuentren caracteres como letras, números, signos, etc., y entender que tipo de carácter se trata.

Para ello se divide la imagen en sectores y se procede a realizar un sistema de tratamiento de imágenes.

Primero dividimos la imagen en líneas y luego en sectores, como en la siguiente imagen se puede observar:

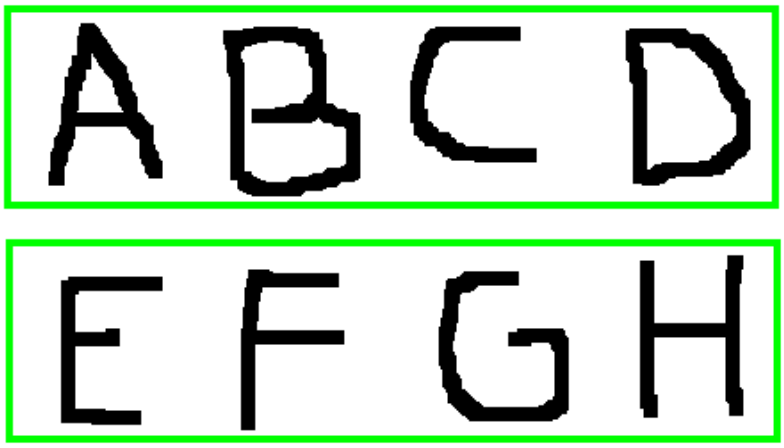


Figura 61. División de caracteres por línea (Imagen tomada del software para visión artificial que se está desarrollando)

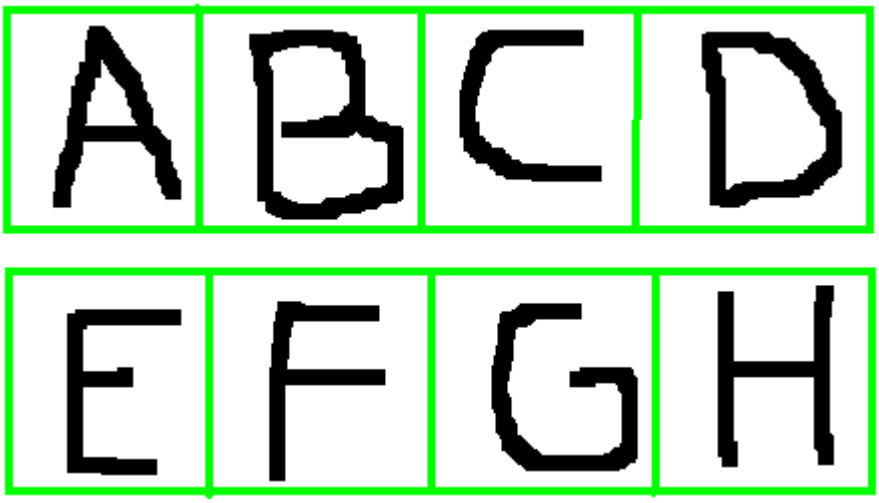


Figura 62. División de caracteres por sección (Imagen tomada del software para visión artificial que se está desarrollando)

Después de dividir los caracteres se recurre a llamar a una plantilla y matriz de imágenes para reconocer que tipo de letra o carácter es el seleccionado en la imagen.

En el diseño de reconocimiento óptico de caracteres que se realizo se incluyo un algoritmo de precisión en el momento de reconocer la palabra

NEGRO y BLANCO, así existan manchas o las letras sean no tan precisas en la imagen.

El control de la cinemática y del envió de los datos al robot se vio en los capítulos anteriores y se los puede observar en el ANEXO 2 donde está el programa realizado en matlab con sus respectivos comentarios.

4.2.3 Reconocimiento de color y tratamiento de imágenes

Para leer imágenes contenidas en un archivo al ambiente de matlab se utiliza la función imread, cuya sintaxis es:

imread('nombre del archivo')

Donde nombre del archivo es una cadena de caracteres conteniendo el nombre completo de la imagen con su respectiva extensión, los formatos de imágenes que soporta matlab son:

TABLA 11. TABLA DE DE EXTENSIÓN DE IMÁGENES QUE LEE MATLAB, TOMADO DE INTERNET

Formato	Extensión
TIFF	.tiff
JPEG	.jpg
GIF	.gif
BMP	.bmp
PNG	.png
XWD	.xwd

Representación de una imagen en escala de grises y RGB en matlab es atreves de las siguientes matrices.

Ecuación en escala de grises:

$$I(x,y)\begin{bmatrix} v_{11} & v_{12} & v_{1v} \\ v_{12} & v_{22} & v_{2n} \\ v_{m1} & v_{m2} & v_{mn} \end{bmatrix}$$ **Ecuación** [4.7]

Ecuación en RGB:

$$I_R(m,n,1)\begin{bmatrix} r_{11} & r_{12} & r_{1v} \\ r_{12} & r_{22} & r_{2n} \\ r_{m1} & r_{m2} & r_{mn} \end{bmatrix} I_G(m,n,2)\begin{bmatrix} g_{11} & g_{12} & g_{1v} \\ g_{12} & g_{22} & g_{2n} \\ g_{m1} & g_{m2} & g_{mn} \end{bmatrix} I_B(m,n,3)\begin{bmatrix} b_{11} & b_{12} & b_{1v} \\ b_{12} & b_{22} & b_{2n} \\ b_{m1} & b_{m2} & b_{mn} \end{bmatrix}$$

Ecuacion [4.8]

Si se quisiera introducir la imagen contenida en el archivo data.jpg a una variable para su procesamiento en matlab, entonces se tendría que escribir en línea de comandos: > > imagen = imread (’data.jpg’);

Una función que permite encontrar el tamaño de la imagen es size (variable)

[m, n]=size(image); con esta función encontramos el tamaño de pixeles en m y n se guardan los valores.

Para grabar el contenido de una imagen en un archivo se utiliza la función imwrite (variable,’nombre del archivo’), en donde variable representa la variable que contiene a la imagen y nombre del archivo el nombre del archivo con su respectiva extensión.

Después que realizamos un procesamiento con la imagen, es necesario desplegar el resultado obtenido, la función imshow(variable) permite desplegar la imagen en una ventana en el ambiente de trabajo de matlab.

imshow(imagen).

Para tener el numero de pixel en matlab y saber qué valor corresponde se utiliza lo siguiente, pasamos la imagen a escala de grises, tomamos su valor con la función size que vimos anteriormente, y luego escribimos imagen (pixelx,pixely) es decir el punto y obtendremos un valor.

El toolbox de matlab nos da una infinidad de herramientas para el manejo de la visión artificial y solo hay que saber utilizarlo.

4.2.4 Filtraje espacial

Tomado Procesamiento digital de imágenes (Carlos Platero) y del ensayo de Visión Artificial (Universidad Nacional de Quilmes – Ing. en Automatización y Control Industrial). El filtraje espacial es una de las operaciones comunes en la visión computacional ya sea para realizar efectos de eliminación de ruido o bien detección de bordes. En ambos casos la determinación de los píxeles de la nueva imagen depende del píxel de la imagen original y sus vecinos. De esta forma es necesario configurar una matriz (mascara o ventana) que considere cuales vecinos y en qué forma influirán en la determinación de el nuevo píxel. Consideremos una imagen IS (m; n).

Para desarrollar en matlab este tipo de operaciones se utiliza la función nlfilter, cuya estructura es la siguiente:

IT=nlfilter(IS,[i j],fun);

Donde IT es la variable que contiene la imagen resultado de la operación, IS es la variable que contiene a la imagen original, [i j] son las dimensiones de la máscara que define la influencia de los vecinos para el cálculo del nuevo píxel por ultimo fun representa la función que desarrolla el cálculo sobre los elementos de la vecindad definidos de dimensión i x j.

La función fun recibe como entrada una matriz x de i x j datos correspondientes a los vecinos de la imagen los cuales son procesados por la función devolviendo el valor que corresponde al dato centrado en la máscara.

4.2.5 Funciones para la extracción de bordes

En visión computacional es de utilidad para hacer reconocimiento de objetos o bien para segmentar regiones, extraer los bordes de objetos (que en teoría delimitan sus tamaños y regiones). La función edge da la posibilidad de obtener los bordes de la imagen. La función permite encontrar los bordes a partir de dos diferentes algoritmos que pueden ser elegidos, canny y sobel. El formato de esta función es: Imagen= edge (ImageS, algoritmo);

ImageR=edge(imagegray,canny);

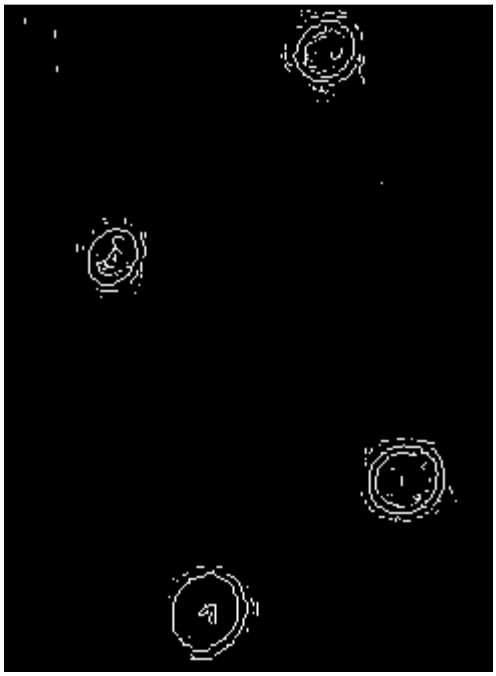


Figura 63. Obtención de bordes en matlab. (Imagen tomada del software para visión artificial que se está desarrollando)

4.2.6 Imágenes binarias y segmentación por umbral.

Una imagen binaria es una imagen en la cual cada píxel puede tener solo no de dos valores posibles 1 o 0. Como es lógico suponer una imagen en

esas condiciones es mucho más fácil encontrar y distinguir características estructurales.

En visión computacional el trabajo con imágenes binarias es muy importante ya sea para realizar segmentación por intensidad de la imagen, para generar algoritmos de reconstrucción o reconocer estructuras.

Por ejemplo si de la imagen quisiera realizarse este tipo de operación de tal forma que los píxeles mayores a 128 sean considerados como 1 y los que son menores o iguales a 128 como cero, se escribiría en línea de comandos como: result=imagen>=128;

4.2.7 Operaciones morfológicas

Una de las operaciones más utilizadas en visión sobre imágenes previamente binarizadas es las operaciones morfológicas. Las operaciones morfológicas son operaciones realizadas sobre imágenes binarias basadas en formas. Las principales operaciones morfológicas son la dilatación y la erosión. La operación de dilatación adiciona píxeles en las fronteras de los objetos, mientras la erosión los remueve. En el caso de la dilatación, si alguno de los píxeles de la rejilla configurados como unos coincido con al menos uno de la imagen original el píxel resultado es uno. Por el contrario en la erosión todos los píxel de la rejilla configurada como unos deben coincidir con todos los de la imagen si esto no sucede el píxel es 0.

En matlab las funciones utilizadas para realizar estas dos operaciones morfológicas son erode y dilate. El formato de ambas funciones es:

ImageR=erode (ImageS,w);

ImageR=dilate (ImageS,w);

Donde ImageR es la variable que recibe a la imagen resultado, ImageS es la imagen binaria origen a la que se desea aplicar la operación morfológica y w

es una matriz de unos y ceros que determina el formato y estructura de la rejilla.

4.2.8 Selección de objetos

En visión por computador resulta de especial utilidad de poder aislar objetos de una imagen binaria con un método rápido e interactivo. La función de matlab bwselect permite interactivamente seleccionar el objeto binario a segmentar con tan solo señalarlo en la ventana (previamente desplegada mediante la función imshow).

Así mismo podemos encontrar diferentes tipos de objetos en una imagen usando las siguientes instrucciones:

imagen=imread('c:\foto.jpg');

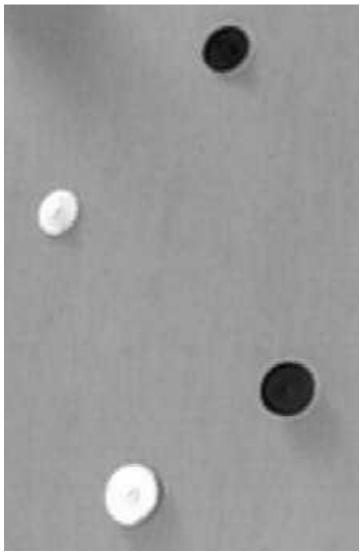


Figura 64. Abrir una imagen con inshow. (Imagen tomada del software para visión artificial que se está desarrollando)

imagengris= **rgb2gray** (imagen);

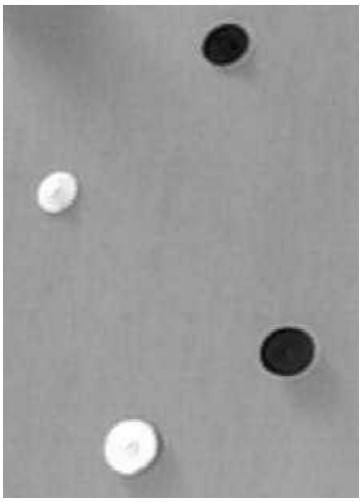


Figura 65. Abrir una imagen con inshow en escala de grises. (Imagen tomada del software para visión artificial que se está desarrollando)

imagenedge= **edge** (imagengris, 'prewitt');

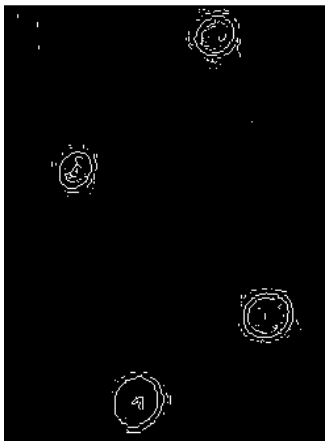


Figura 66. Abrir una imagen con inshow en busca de sus bordes delgados. (Imagen tomada del software para visión artificial que se está desarrollando)

imagendila= **imdilate**(imagenedge,SE)

Procedimos a dilatar los bordes para mejor trabajo de selección de áreas.

imagen1= **not** (imagendila)

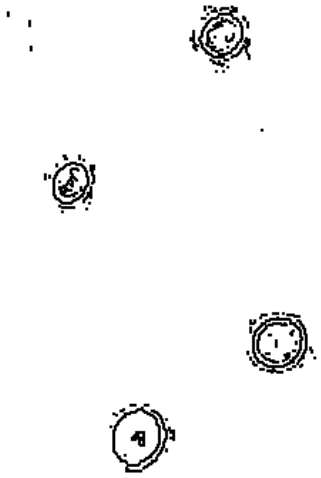


Figura 67. Abrir una imagen con inshow de borde, pero con colores invertidos. (Imagen tomada del software para visión artificial que se está desarrollando)

En este punto toca contar cuantos elementos se encuentran en esta imagen para ello utilizamos la función [L Ne]=bwlabel(imagen1), donde Ne es el número de elementos en la imagen.

Ahora necesitamos conocer las propiedades de cada elemento en la imagen para ello utilizamos la función propiedades= regionprops(L,'basic'); donde en propiedades se guardaran los datos de que elementos se encuentran ahí, como su área su centroide etc.

Con esta función ya podemos conocer las áreas de los elementos en una imagen y eliminar estos elementos según nuestras necesidades como por ejemplo un tamaño determinado o un color determinado.

Y utilizando un seleccionador en forma de rectángulo contaremos nuestros elementos en la imagen, para ello se utiliza la función:

```
rectangle('Position',propiedades(n).BoundingBox,'EdgeColor','g','LineWidth',4)
```

Que creara al rededor de cada objeto un rectángulo de color verde con una línea de grosor de 4.

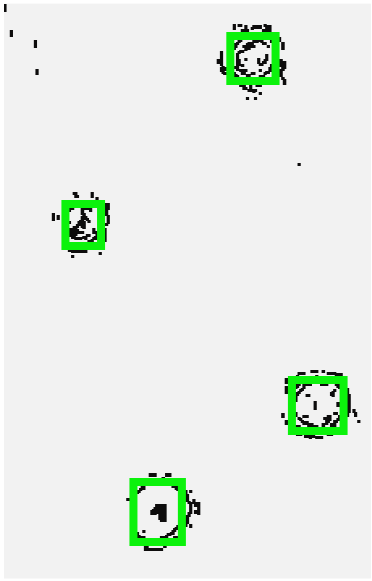


Figura 68. Abrir una imagen con inshow de los elementos necesarios en la imagen. (Imagen tomada del software para visión artificial que se está desarrollando)

En este caso son 4 elementos dentro de nuestra imagen los cuales pueden ser medidos y analizados uno por uno.

4.3 Diseño Electrónico.

Se analizo la propuesta de crear un robot manipulador con visón artificial, el cual esté dispuesto a trabajar autónomamente y controlado por un PC, para ello se diseñara un sistema de control electrónico el cual nos despliegue un MENU de opción, en un LCD, para escoger la forma de trabajo del prototipo.

Para ello la tarjeta electrónica necesaria será principalmente encargada de enviar y recibir datos los cuales serán nuevamente enviados, para así seguir un ciclo de control con interfaz con la PC y un sistema autónomo desde una caja de control. Para ello analizamos que todos los elementos electrónicos

se deben comportar de una manera igual es decir enviar y transmitir señales.

El programa de diseño y simulación electrónico que vamos a utilizar es PROTEUS el cual es un producto de muy alta calidad y de entrono grafico que me permite simular y diseñar correctamente un circuito eléctrico.

Isis es la versión de PROTEUS que es encargada del entorno grafico de circuitos y electrónica.

4.3.1 Conexión serial PIC 16F628A

En esta parte se diseña las conexiones para el funcionamiento del PIC16F628A, que es el encargado de controlar cada servomotor. Como existen 3 servomotores en el prototipo deberá haber 3 tarjetas de control 16F628A, ya que con esto podemos mantener activados y con torque nuestros servomotores:

Primeramente trabajaremos con tensión de la fuente la cual se la debe bajar para poder polarizar nuestro PIC, la fuente utilizada es de 12V, se tendrá un regulador de voltaje IC 7805 que nos permite bajar el voltaje a 5 voltios con lo que el pic trabaja correctamente.

Utilizamos resistencias de 1K para las entradas al pic para bajar la corriente y evitar que se queme algún pin.

El PIC utiliza un cristal de 4MHZ, ya que nuestro sistema no es de presión de tiempos podemos utilizar sin ningún inconveniente este pic a esa velocidad.

En la **Figura 69.** Se puede observar la conexión para comunicación serial. (Imagen tomada del diseño electrónico que se está desarrollando)

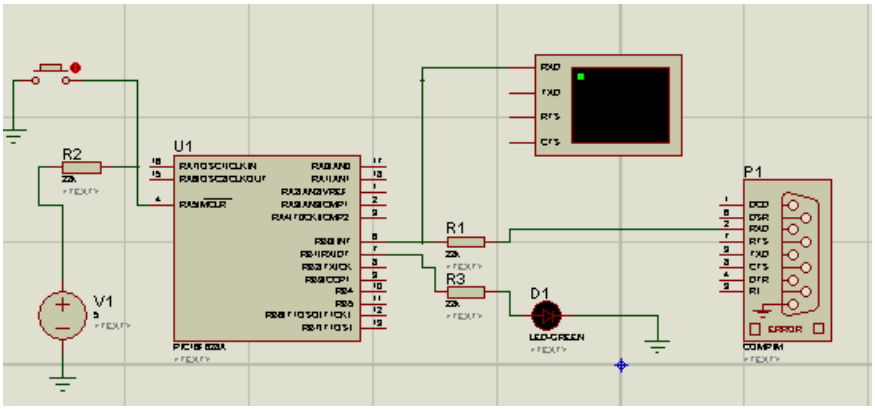


Figura 69. PIC 16f628A conexión serial al PC

También la librería modedefs.bas. Tiene una función la cual es escribir o enviar un dato vía serial atreves del pic por uno de sus puertos.

Envía uno ó más datos, en formato standard asincrónico usando 8 bits de datos, sin paridad y 1 stop bit (8N1) .SEROUT es similar al comando Serout de BS1 .Pin es automáticamente colocado como salida. Pin puede ser una constante, 0 - 15 , ó una variable que contenga un número de 0-15 (p.ej. B0) ó un número de Pin (p.ej. PORTA.0)

Los nombres Mode (p.ej. T2400) están definidos en el archivo MODEDEFS.BAS.

En la **Figura 70.** se observa la conexión de la fuente al IC 7805 para bajar el voltaje y poder polarizar el PIC 16f628A.

El puerto A.0 es la salida del pic hacia el servomotor, con salida de ancho de pulso PWM, en este PIN pondremos una bornera para conexión vía cable.

El puerto B.0 es la entrada que se utilizara como receptor de las señales seriales enviadas por el PIC MASTER o el PC, en este PIN pondremos una bornera para conexión vía cable.

Además se utilizaran 3 de estas tarjetas de circuitos ya que en el robot se utilizaran 3 servomotores.

4.3.2 Diseño HW del circuito de control.

La característica principal de los robots es su programabilidad, lo que permite modificar su funcionamiento de acuerdo a la información que recibe de su entorno, y con los avances en la tecnología del procesamiento de información, los robots son mas autónomos y su potencial es cada vez mayor.

4.3.2.1 Modo autónomo.

En la parte autónoma y digital de nuestro control es cuando vamos a poder mover nuestro brazo robótico, de un lado a otro y encender el efector final utilizando controles físicos como son los pulsadores.

Además en la caja de control se colocaron borneras tipo banana para poder interactuar con otros microcontroladores externos.

Utilice 9 pulsadores cada uno conectado al pic de manera que se pueda trabajar con la tierra es decir cuando se presione un pulsador el pic recibirá un cero lógico, o a su vez si un microcontrolador o señal externa envía un uno lógico por las bornera el PIC de control recibirá un cero lógico.

Los pulsadores se encuentran conectados en el puerto C y un pulsador en el puerto B.6.

Portc.0 = Pulsador 1 “controla el menú del LCD”

Portc.1 = Pulsador 2 “controla el menú del LCD”

Portc.2 = Pulsador 3 “controla el movimiento a la derecha del brazo uno”

Portc.3 = Pulsador 4 “controla el movimiento a la izquierda del brazo uno”

Portc.4 = Pulsador 5 “controla el movimiento a la derecha del brazo dos”

Portc.5 = Pulsador 6 “controla el movimiento a la izquierda del brazo dos”

Portc.6 = Pulsador 7 “controla el movimiento a la derecha efector final hacía arriba”

Portc.7 = Pulsador 8 “controla el movimiento a la derecha efector final hacía abajo”

Portb.6 = Pulsador 9 “controla la activación y desactivación del electroimán del efector final”

Cada vez que se presiona un pulsador de movimiento una señal es enviada al pic 16F628A correspondiente vía serial por el puerto B.0 B.1 B.2, respectivamente hacia el brazo uno, brazo dos, efector final.

El LCD permite visualizar un menú el cual tiene las opciones de:

1. **Autónomo**
    - Digital
    - Análogo
2. **Controlado**

Este menú se lo puede controlar con los pulsadores de control de menú.

El relé se activa cada vez que se presiona el pulsador 8, se activa atreves de la salida del PIC del PuertoE.2, mediante una resistencia se activa el Tip122 el cual cierra el circuito ya que solo trabajo como switch y el relé cambia su posición, es decir transmite 12 voltios hacia el electroimán que está en la punta del efector fina.

La parte de conexión al relé es también llamada conexión de potencia puesto que subimos el voltaje para poder activar un electroimán.

En esta parte trabaje con las entradas análogas del PIC del puerto A, las dos entradas análogas del puerto son:

PortA.0 = entrada análoga uno “Controla el movimiento del brazo uno”

PortA.1 = entrada análoga dos “Controla el movimiento del brazo dos”

Y con el digital de control o Pulsadores 6, 7 y 8 controlamos el efector final

4.3.2.2 Modo Controlado.

En esta parte se le indica al controlador que se debe conectar vía serial al PC, con lo cual se transmitirán datos del PIC al PC, para el control automático del robot, el programa realizado en MATLAB para el control del PC se explicara en el siguiente tema.

Los puertos utilizados para enviar y recibir datos son PUERTO B.5 y PUERTO B.7 los cuales se conecta aun DB9 el cual va al computador y así tenemos una conexión serial para enviar y recibir datos, además para llegar a tener una conexión exitosa y evitar problemas debemos agregar un IC

MAX 232 el cual transforma la señal TTL en señal CMOS. Es decir cambia de voltaje de 5 V del PIC a 12 del PC.

Para salir de esta parte se lo controla desde un programa visual en la PC.

Diseño de la tarjeta electrónica electrónico

La tarjeta se diseño en una baquelita, de un solo lado de cobre para impresión, de medidas de 13cm X 13cm, la cual fue fundida en acido ferrico a una temperatura de 70 grados centígrados.

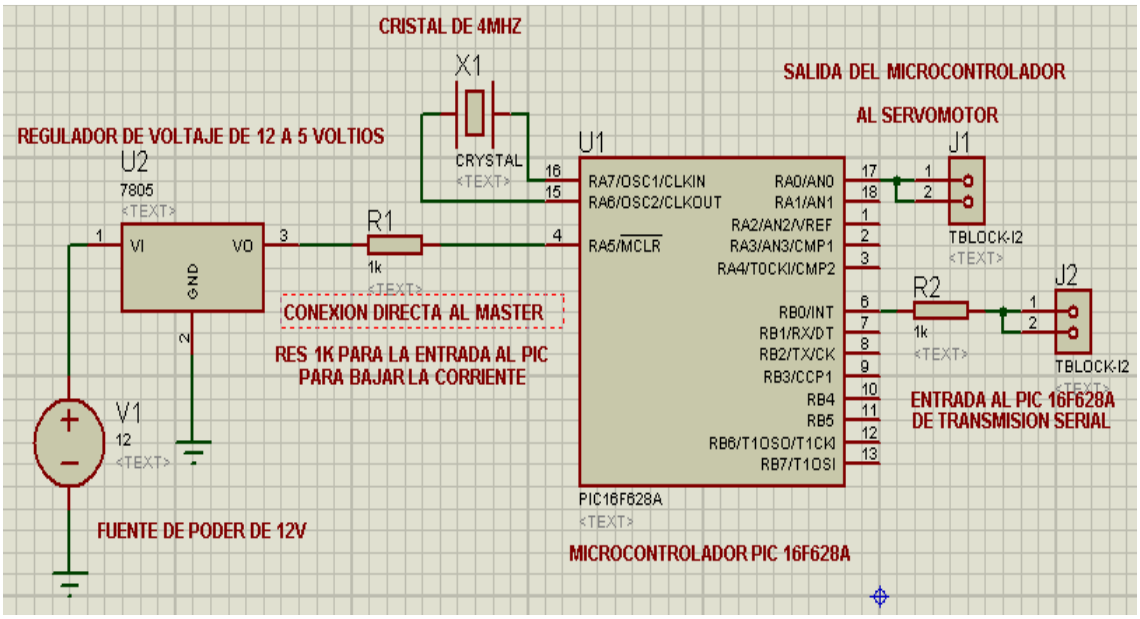


Figura 70. Imagen de la conexión el PIC 16F628A para transmisión y recepción de datos en serie.

(Imagen tomada del diseño electrónico que se está desarrollando)

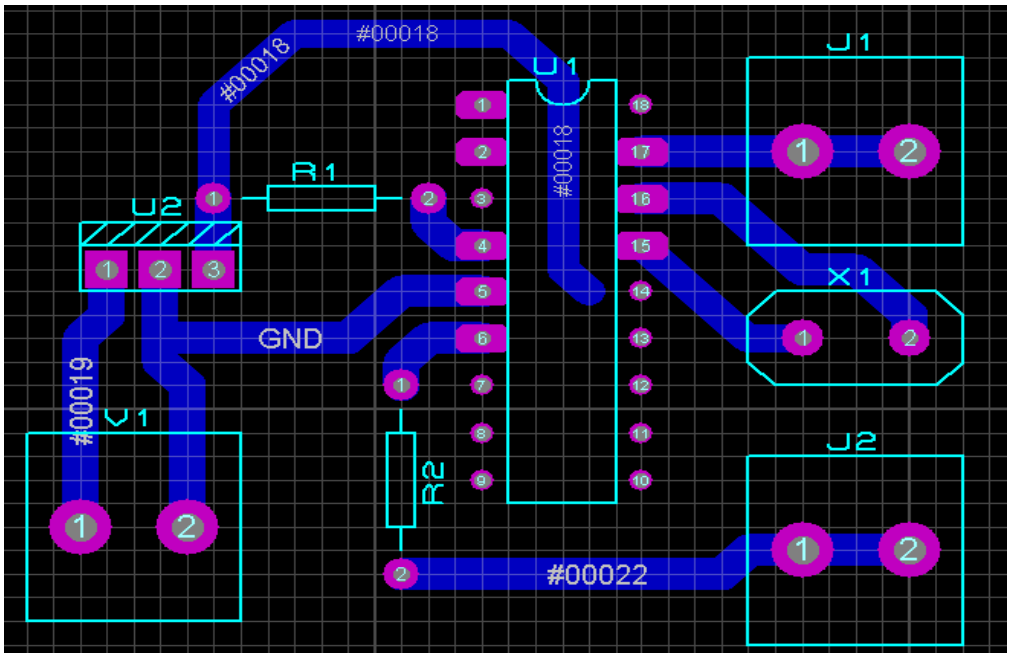


Figura 71. Imagen de la conexión el PIC 16F628A PCB para transmisión y recepción de datos en serie.

(Imagen tomada del diseño electrónico que se está desarrollando)

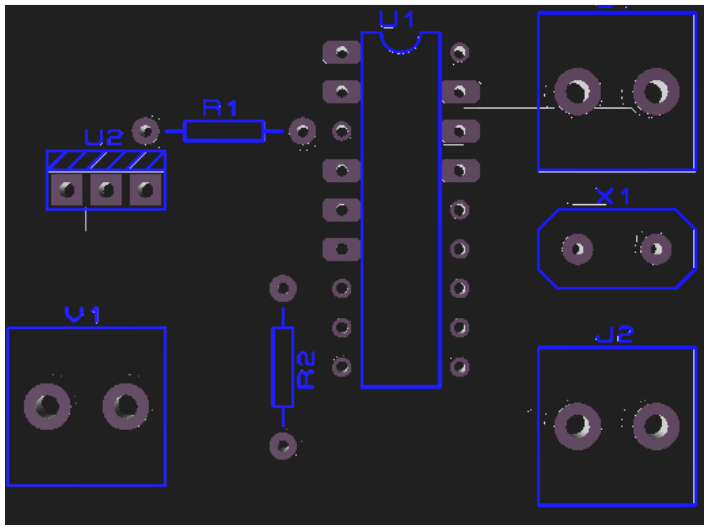


Figura 72. Imagen 3D PCB de la conexión el PIC 16F628A para transmisión y recepción de datos en serie.

(Imagen tomada del diseño electrónico que se está desarrollando)

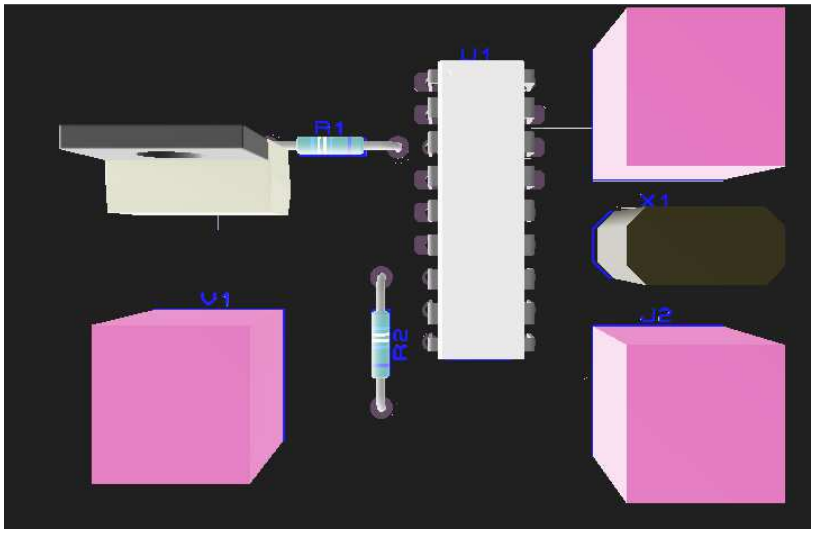


Figura 73. Imagen 3D de la conexión el PIC 16F628A para transmisión y recepción de datos en serie.

(Imagen tomada del diseño electrónico que se está desarrollando)

4.3.3 Conexión serial PIC 16F877A.

En la **Figura 74.** Se observa cómo se utiliza el IC 7805 para bajar el voltaje y poder polarizar el PIC 16F877A, además de su conectar un cristal de 4Mhz.

Los Pines 2 y 3 correspondientes a las entradas análoga poarta.0 y porta.1 están conectadas a un filtro pasa bajos para que la señal que entre sea lo más limpia posible.

El puertoE.2 es una salida para la conexión de un relé el cual será el encargado de activa el electroimán.

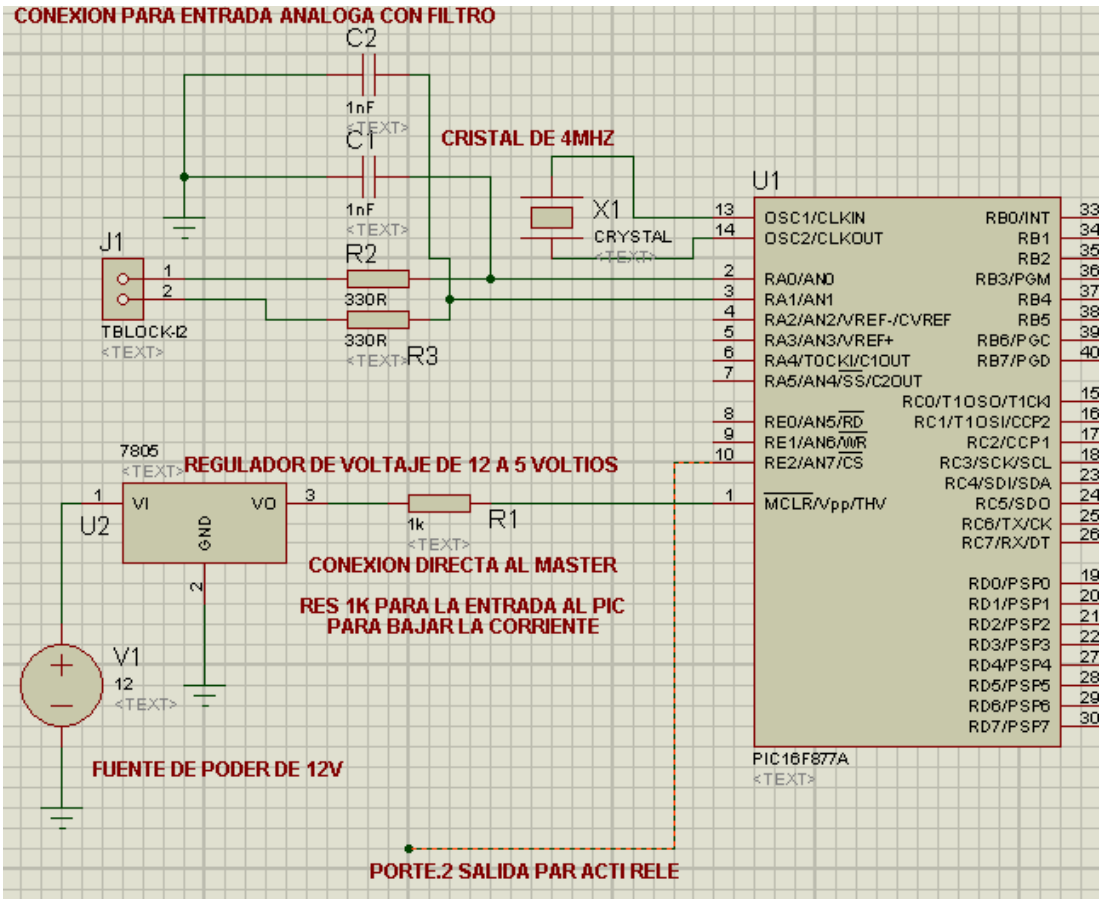


Figura 75. Imagen 1 de la conexión el PIC 16F877A para control.

(Imagen tomada del diseño electrónico que se está desarrollando)

Trasmisión de datos del PIC 16F877A al LCD.

En la **Figura 76.** Se observa la mitad del diseño en la que el PIC se conecto al LCD y el puerto C son las entradas para recibir los datos de control Autónomo del PIC, también se ve que el puerto B será considerado más adelante como salidas y entradas.

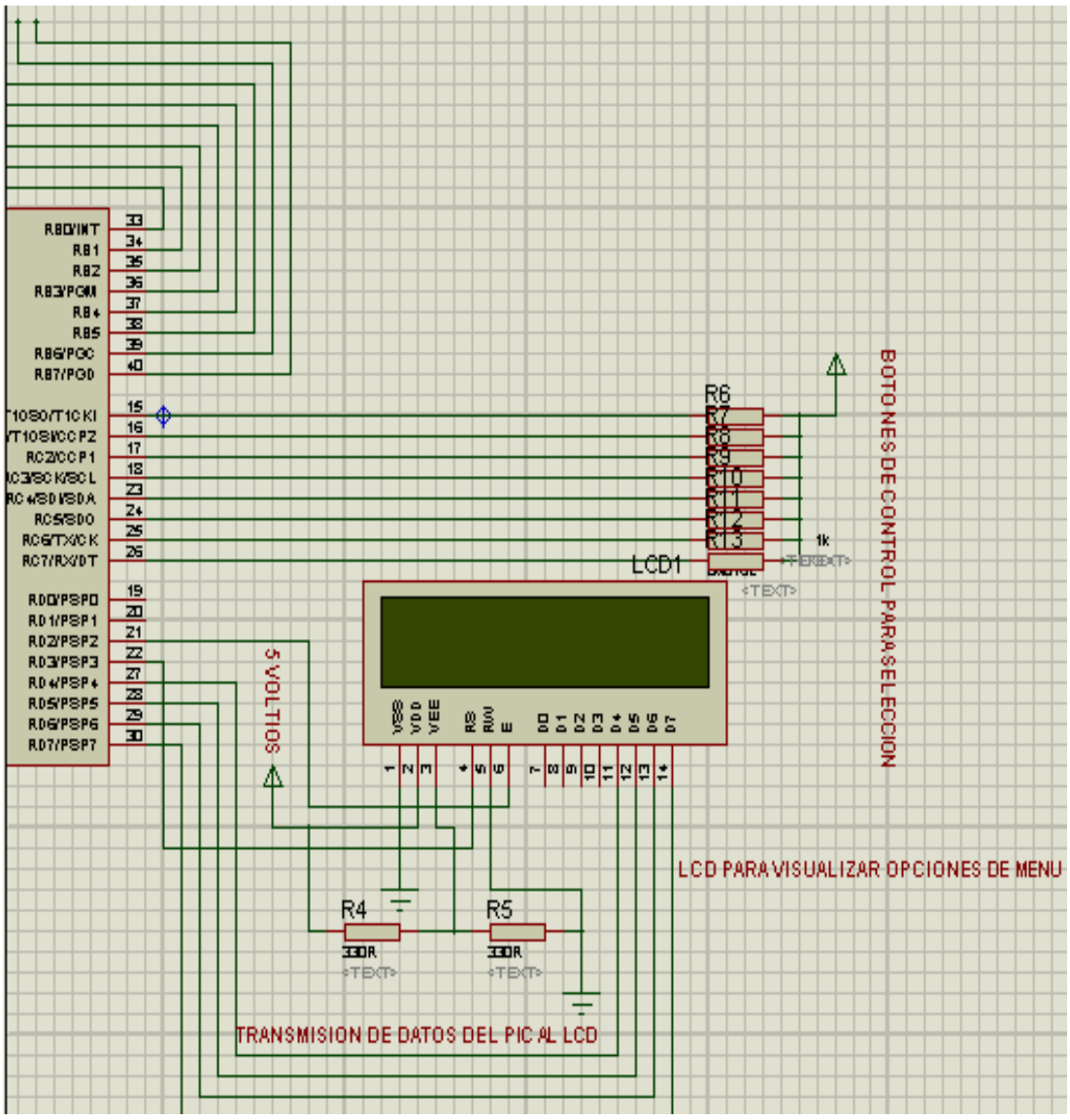


Figura 76. Imagen 2 de la conexión el PIC 16F877A para control.

(Imagen tomada del diseño electrónico que se está desarrollando)

Conexión de los tips 122.

En esta **Figura 77.** Podemos observar la conexión de los tip122 para pasar de estado en el pic, estos están conectados directamente como switch, también los botones del lado izquierdo indican un cambio en el estado del Pic.

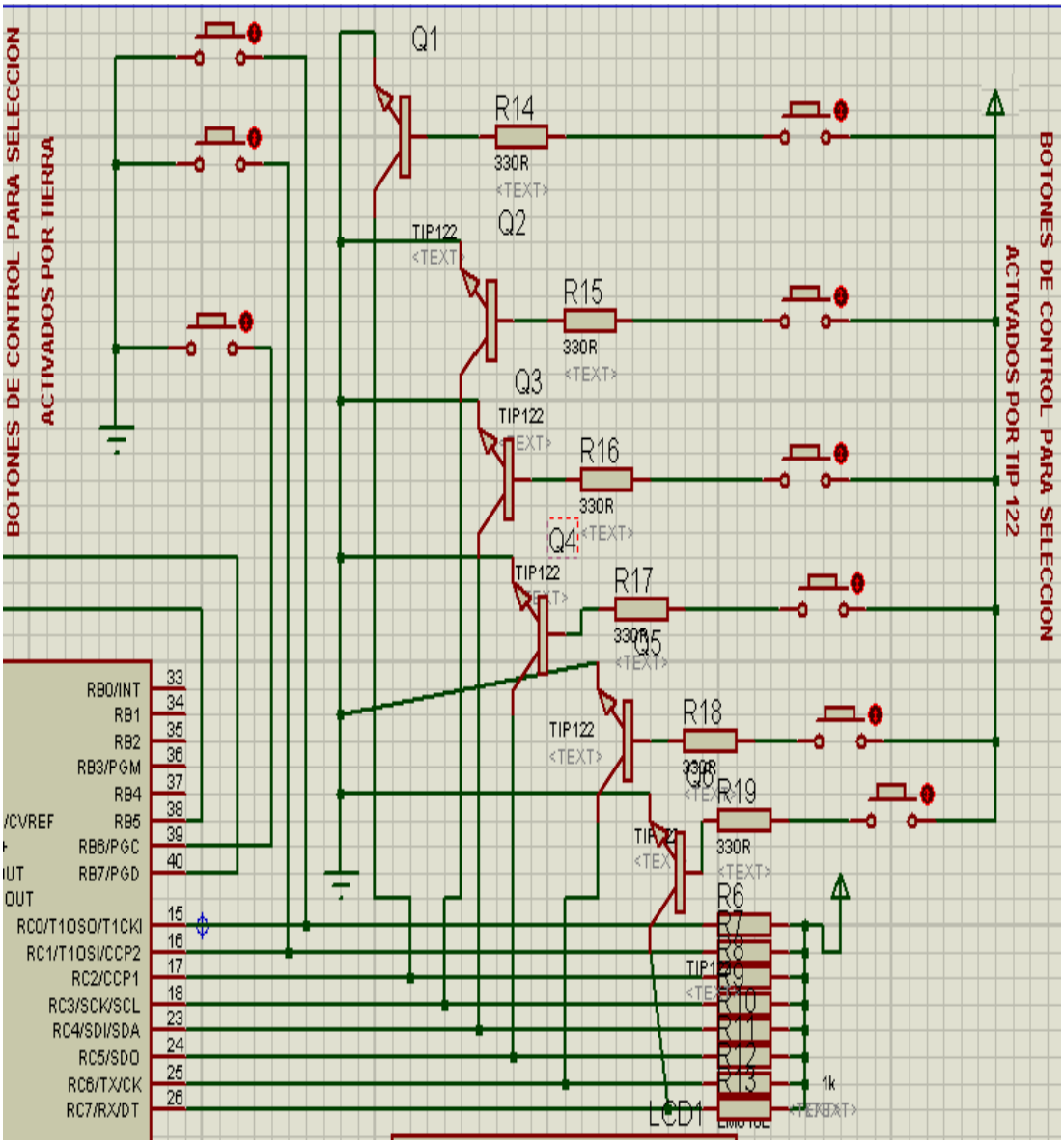


Figura 77. Imagen 3 de la conexión el PIC 16F877A para control.

(Imagen tomada del diseño electrónico que se está desarrollando)

Conexión para entrada analógica.

En la **Figura 78.** Observamos la conexión del PIC 16F877A aun BD9 el cual será la conexión directa al PC para comenzar a trabajar como sistema controlado. Atraves del puerto B.7 y B.5

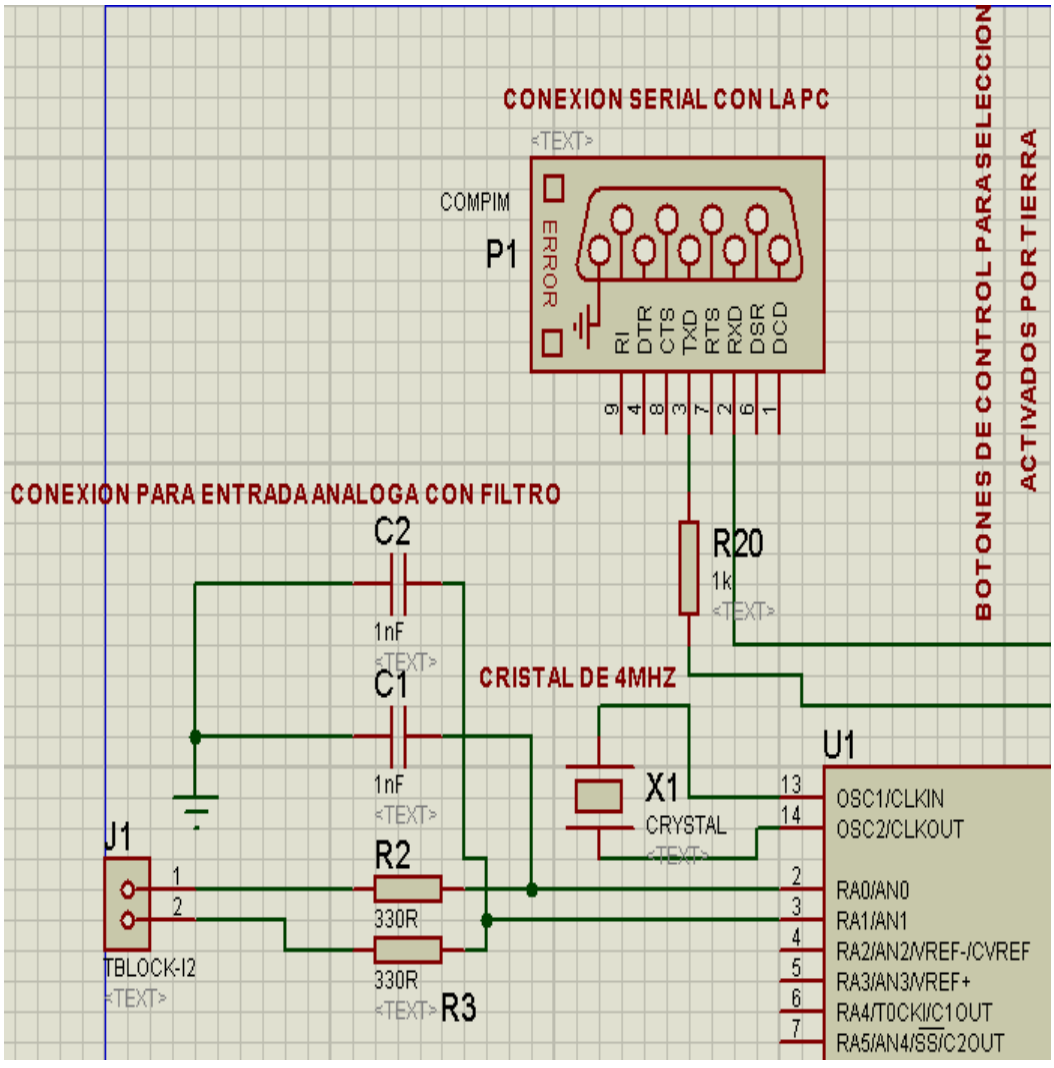


Figura 78. Imagen 4 de la conexión el PIC 16F877A para control.

(Imagen tomada del diseño electrónico que se está desarrollando)

Conexión directa al máster.

En **Figura 79**. Observamos cómo es la conexión de la salida del puerto E.2 para poder accionar un relé para la activación del efector final.

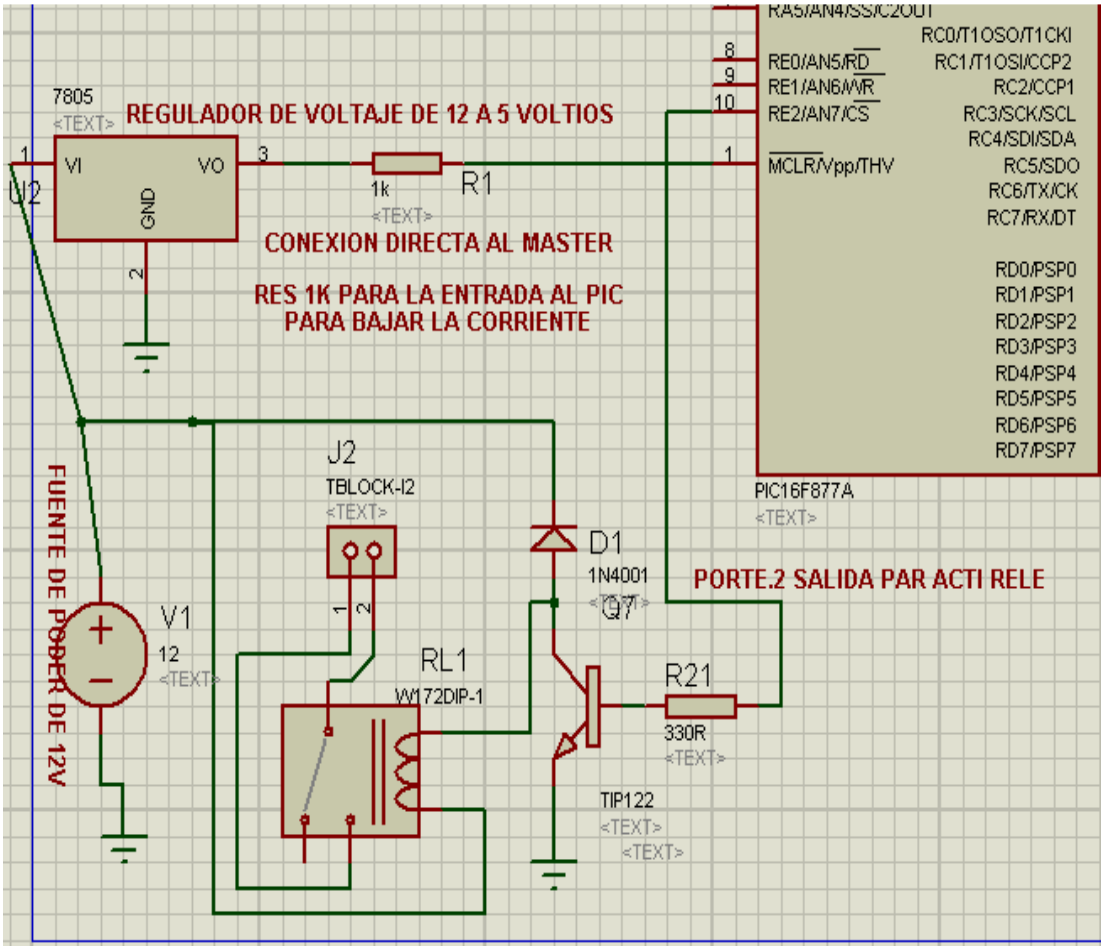


Figura 79. Imagen 5 de la conexión el PIC 16F877A para control.

(Imagen tomada del diseño electrónico que se está desarrollando)

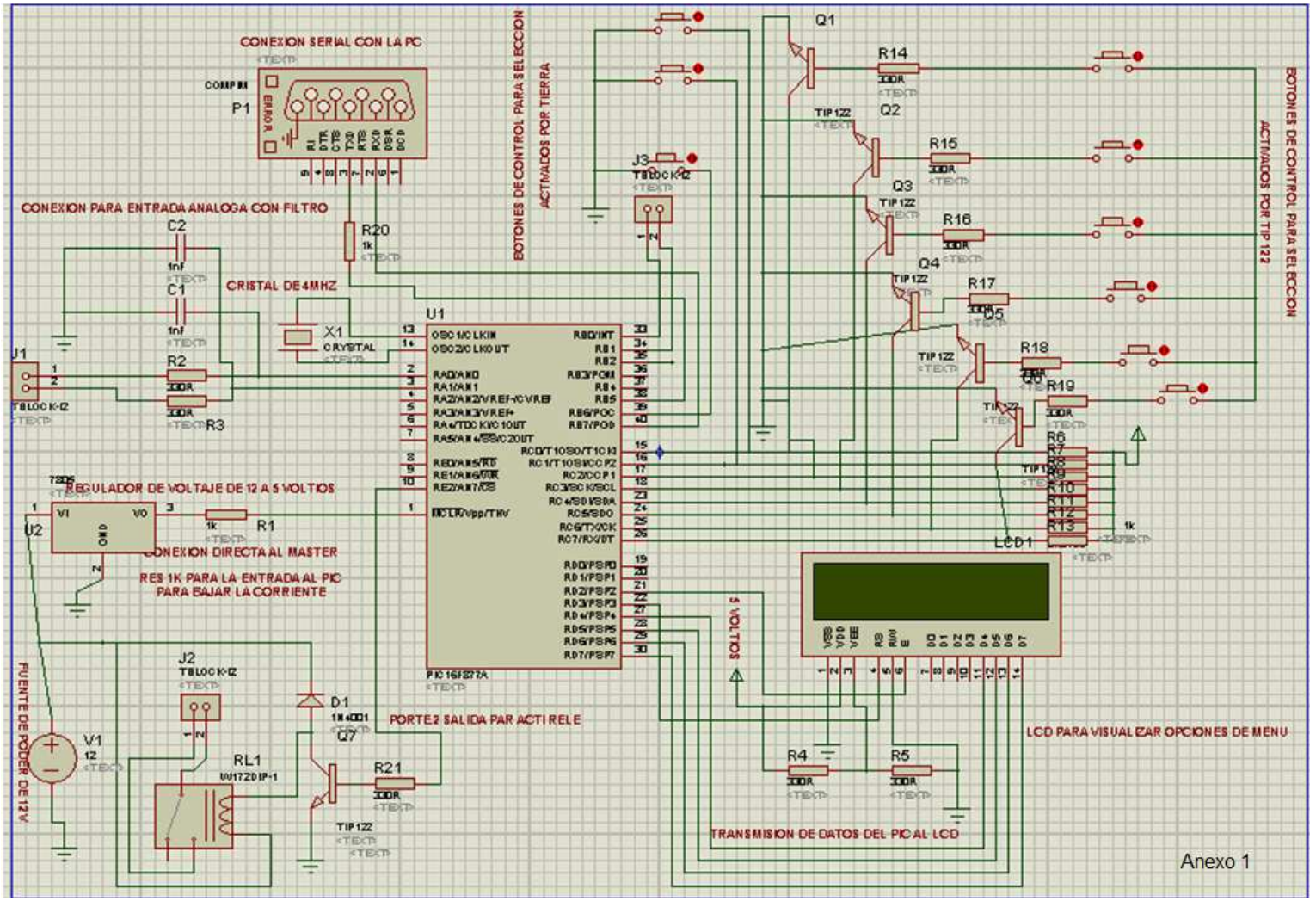


Figura 80. Imagen de la conexión el PIC 16F877A para control. (Imagen tomada del diseño electrónico que se está desarrollando)

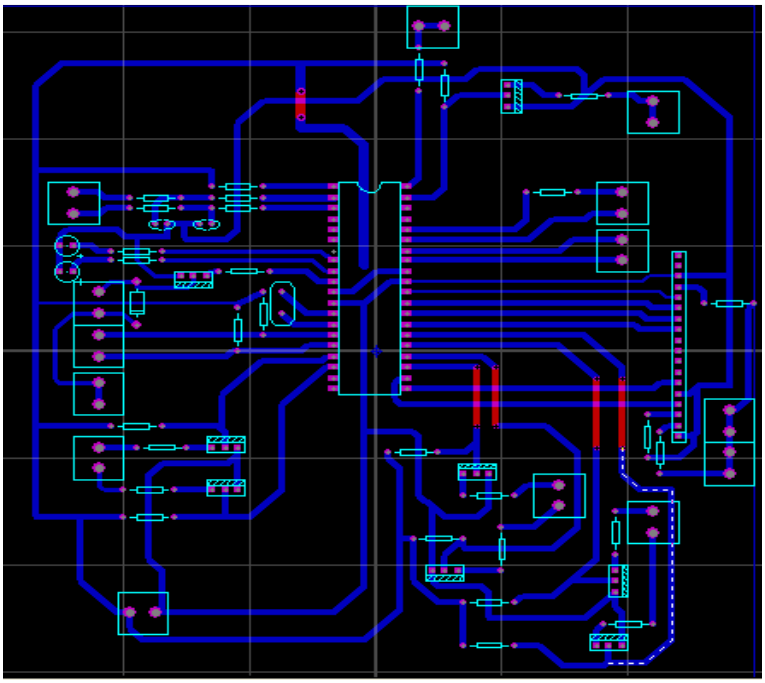


Figura 81. Diseño PCB de tarjeta de control PIC 16F877A

(Imagen tomada del diseño electrónico que se está desarrollando)

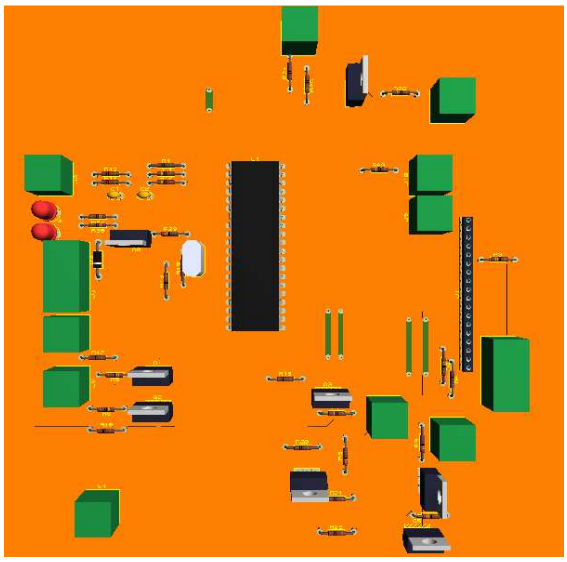
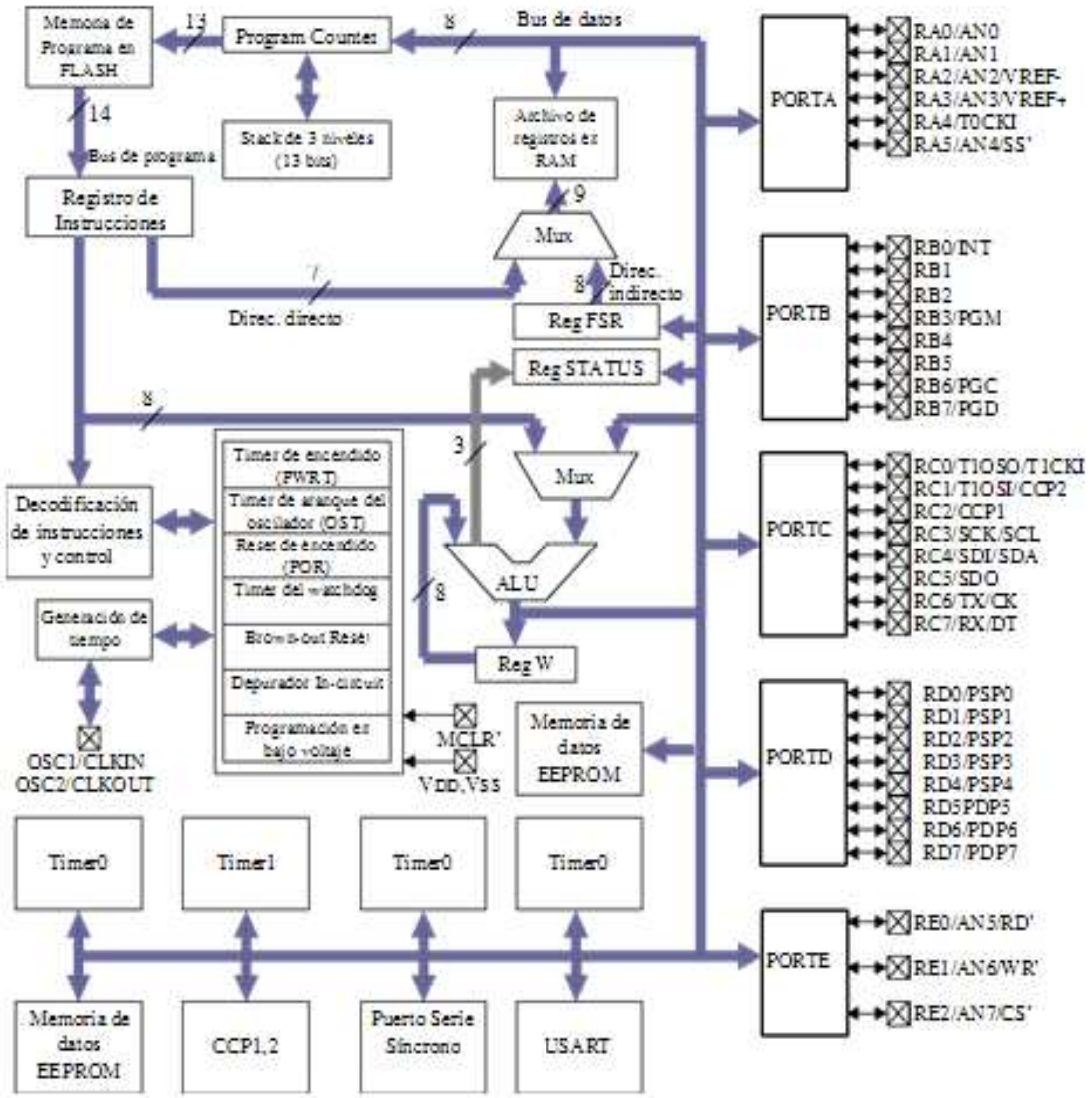


Figura 82. Diseño 3D de tarjeta de control PIC 16F877A

(Imagen tomada del diseño electrónico que se está desarrollando)



.

Figura 83. Bloques la organización interna del PIC16F877, imagen tomada de la hoja de datos del PIC.

(Imagen tomada del diseño electrónico que se está desarrollando)

4.4 Diseño de SW

La programación del pic se desarrollo en PIC BASIC PRO y este es el seudoprograma para llegar a la solución:

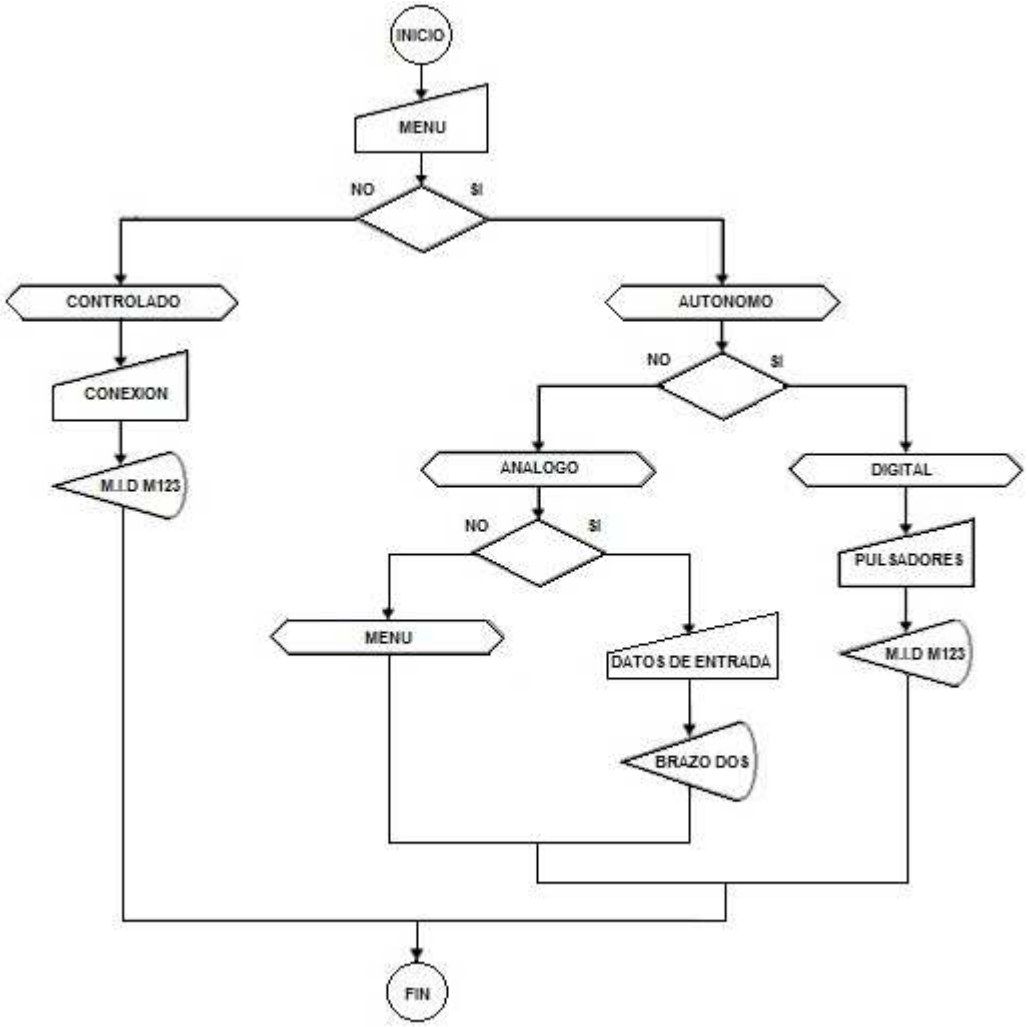


Figura 84. Diagrama de proceso seudoprograma control PIC 16F877A

En la primera parte del diagrama tenemos el inicio donde se declaran las variables y se llaman a las librerías necesarias para realizar un programa de control para este robot. En la parte de menú es el segundo cuadro que tenemos en el diagrama de flujo es una entrada para elegir las dos opciones de menú que declaramos en el programa, a través de un cuadro de condición, opciones 1 y 2, controlado y autónomo.

En el Diagrama, MENU autónomo, hay como escoger 2 opciones se describe la división del menú de la parte autónomo, en análogo y digital, mediante un cuadro de condición, en el programa se pide que si el PULASADOR 1 es activado vamos a menú digital, y si el PULSADOR 2 es activado vamos análogo.

El programa análogo, tenemos 2 opciones ingresar datos o regresar al menú, en el momento que ingresan los datos análogos por ejemplo de un potenciómetro a 5 voltios, tendremos una salida de un brazo hacia el movimiento del potenciómetro.

La entrada análoga en el programa PBP es el puerto A.0 y A.1.

En caso de querer regresar al menú se utilizara el PULSADOR UNO.

Entrada digital tenemos 7 Pulsadores los cuales están conectados al pic al puerto C y cada vez que este sea un cero lógico tendremos como salidas movimientos del servomotor. De cada servomotor por eso son 7 pulsadores 2 para cada servomotor y uno para activar y desactivar el electroimán.

En caso de querer regresar al menú se utilizara el PULSADOR UNO.

El programa en la conexión a la PC, tenemos como entrada datos que llegan desde la PC y salidas del PIC hacia los controladores de los servomotores que son las tarjetas con el pic 16f628A.

Para ello utilizamos las etiquetas de control y las funciones del PBP para transmisión de datos serialmente, el código de transmisión serial en PBP es SERIN.

Se puede decir que los datos que llegan al PIC 16F877A son procesados y enviados al servomotor que debe moverse el primer dato es del servo de la base el segundo del servo del codo y el tercero del efector final.

En el **ANEXO 1** se tiene el código completo que se desarrollo para la tesis.

4.4.1 Diseño de la caja de control para el robot con visión artificial.

Esta caja se diseño para poder ser un control con interfaz humano, es de muy fácil uso y diseñada para trabajo en el laboratorio, además se puede trabajar con conectores tipo banana los cuales son utilizados en los laboratorios de potencia.

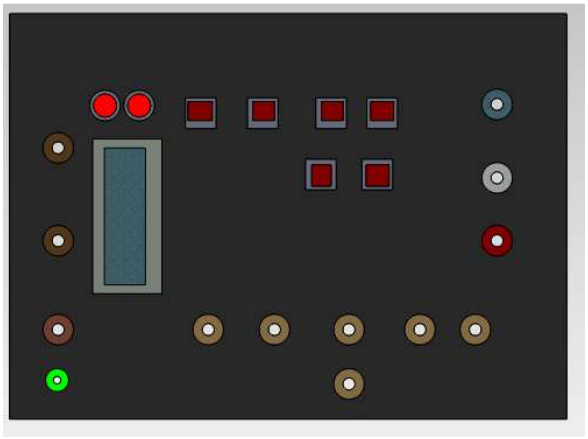


Figura 85. Tapa de control para el manejo del robot vista superior

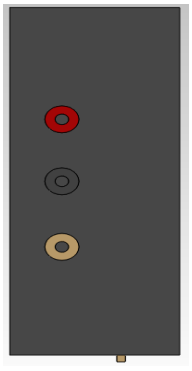


Figura 86. Caja de control para el manejo del robot vista lateral

4.4.2 Descripción de la caja de control para el robot con visión.

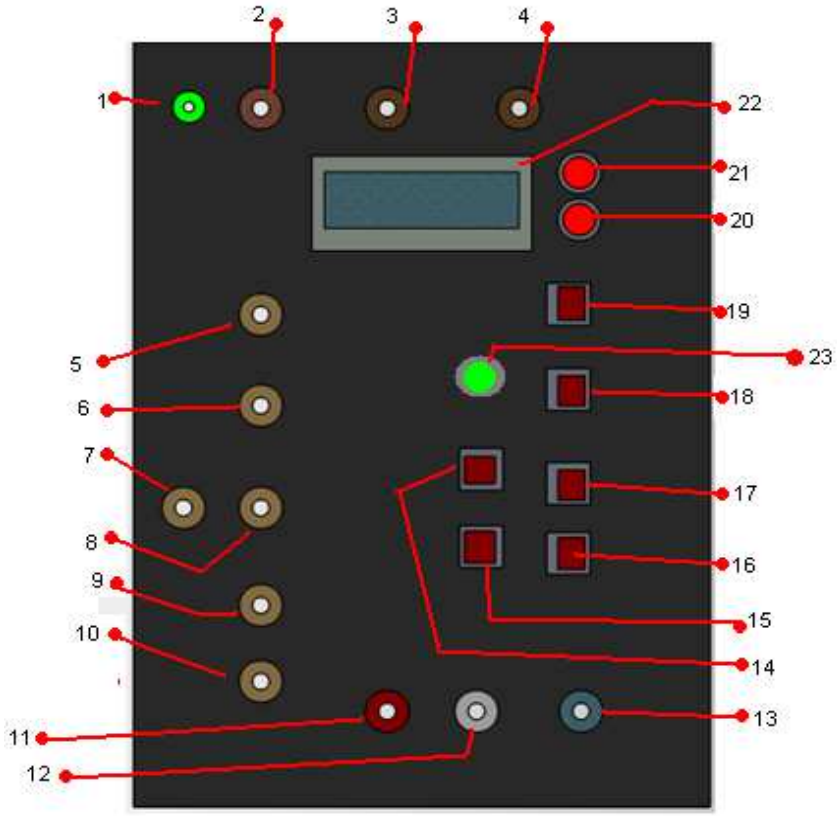


Figura 87. Caja de control ARMADA para el manejo del robot con puntos para la leyenda y descripción.

En la **Figura 87.** podemos ver los controles para el manejo del robot manipulador con visión artificial, y a continuación se describirán para su mejor comprensión:

1. Sensor para activar/desactivar el efector final y electroimán.
2. Salida de 12 V.
3. Análogo 1
4. Análogo 2
5. Bornera para mover brazo 1 derecha.

6. Bornera para mover brazo 1 izquierda.
7. Bornera para mover brazo 2 derecha.
8. Bornera para mover brazo 2 izquierda.
9. Bornera para mover el motor 3 derecha.
10. Bornera para mover el motor 3 izquierda.
11. Salida motor 1.
12. Salida motor 2.
13. Salida motor 3.
14. Mover motor 3 a la izquierda.
15. Mover motor 3 a la derecha.
16. Mover brazo 1 a la derecha.
17. Mover brazo 1 a la izquierda.
18. Mover brazo 2 a la derecha.
19. Mover brazo 2 a la izquierda.
20. Control 2 del menú.
21. Control 1 del menú.
22. LCD para visualizar las operaciones y seleccionar las mismas.
23. Pulsador para activar el electroimán.

4.5 Visón artificial y OCR

Para el diseño del control de visón artificial, principalmente utilizamos de OCR del reconocimiento óptico de caracteres, ya que lo principal es escoger la caja con el nombre del color puesto, es decir si una caja tiene puesto negro en esa caja se colocaran las piezas negras, entonces el OCR es lo principal en la parte de reconocimiento de las palabras y se aumento

su eficiencia y eficacia creando un algoritmo el cual determina al 100% la palabra escrita.

Primeramente para poder realizar la OCR necesitamos crear una base de datos con las formas de las letras en extensión de grafico que acepta MATLAB como por ejemplo JPG, GIF, etc.

Para realizar este proceso son los siguientes pasos:

1) Creamos una imagen con la forma de cada letra y numero.
2) Creamos una matriz con las letras y números
3) Leemos cada imagen y la guardamos en una constante
4) A todas estas constantes las unimos en una matriz.

Con este proceso obtenemos la matriz donde vamos a comparar nuestras palabras obtenidas con la cámara web. Después de crear la matriz vamos a realizar un sistema el cual divida cada palabra en secciones, y cada sección en letras. Se procede a crear la función que divide la imagen en líneas y la llamaremos cada vez que la necesitemos.

Segundo una función que lea cada letra y la separe para después comparar cada letra separada de la imagen con las de la matriz y así obtendremos una palabra. Se procede a desarrollar un algoritmo el cual compara cada palabra con otras palabras y se ajusta a la que más se parezca utilizando un método de aprendizaje con una neurona artificial realizada en un algoritmo. Listo sabemos a qué caja pertenece cada palabra.

Ahora usando tratamiento de imágenes separamos los colores de cada pieza.

Tomamos la imagen real tomada por la webcam y se procede a utilizar el tratamiento de imágenes para poder separar las piezas que se encuentran en la foto estas piezas se deberán separar por colores blanco y negro, y para ello se utilizan lo siguiente:

1) Pasamos la imagen tomada por la webcam y la pasamos a escala de grises

2) La imagen en escala de grises la pasamos por una función edege la cual nos entrega una nueva imagen en blanco y negro con los bordes de la imagen principal resaltados en blanco y el fondo en negro.

3) Creamos una función para dilatar los bordes y tener una imagen un poco más definida.

4) Se invierte el color de la imagen que contiene los bordes para una mejor selección de las piezas para poder eliminar el ruido.

5) Mediante la función bwlabel y un lazo for contamos todas los elementos de la imagen de los bordes y sus propiedades, estos son guardados en dos variables.

6) Creamos una variable de sus propiedades básicas con la función regionprops, esta opción nos entrega las propiedades básicas de cada elemento la imagen, es decir su área, centroide, color, perímetro.

7) Creamos una caja rectangular que rodea a cada elemento de la imagen utilizando la función BoundingBox.

8) Buscamos y eliminamos los objetos que tengan aéreas menores a la de las fichas en la imagen, como es el ruido, imperfecciones de la tabla etc, utilizando la opción find.

9) Tenemos seleccionados los elementos en una caja o recuadro verde

10) Buscamos los centroides de cada elemento utilizando la variable propiedades, nos entrega una información en eje Y, X los cuales son medidas en pixeles.

11) Teniendo el centroide buscamos que color es ese pixel en la imagen, si es cero el valor de la imagen entonces es blanco y si es uno es negro.

12) Ahora ya sabemos los colores de cada pieza

13) Teniendo los centroides y las medidas de la base calculamos las medidas reales de los centroides en la tabla de trabajo, es decir pasamos de pixeles a cm

14) Mediante cinemática y geometría obtenemos las medias de los ángulos para poder llegar a esa pieza.

15) Enviamos los datos hacia la caja de control mediante conexión serial utilizando las funciones de conexión serial.

Para poder calcular el movimiento de cada servomotor para que llegue a su destino se tuvo que, tener en cuenta sus medidas constantes, como el tamaño de los brazos y la medida de los lados del la tabla de trabajo, así podemos transformar la medida de los pixeles a cm, y así poder tener mediadas reales.

La cinemática usada es por geometría la cual utilizando varias funciones podremos realizar los cálculos para obtener los ángulos necesarios, para el movimiento (estas ecuaciones las encontramos en la pagina 80, y en el **ANEXO 2**).

Después de esto se envía los datos serialmente, abriendo el puerto serial y creando las variables para poder controlar la conexión.

En el **ANEXO 2** podemos encontrar el programa que se realizo para el control y la visión artificial con sus respectivas definiciones en cada línea de código.

5. Construcción del prototipo de robot manipulador con visión artificial y Análisis de resultados.

5.1 Resultados obtenidos

En el momento de la construcción nos encontramos con grandes problemas ya que para la realización de un robot manipulador se necesita maquinas y tecnología que sean exactas, pero como es un prototipo se desarrollo de la mejor manera y con la mayor exactitud.

Lo más difícil de la construcción fue el desarrollo de los ejes ya que el peso, del resto del robot estaría sobre estos.

La solución a este problema fue poner una base mas grade y resistente que ayude a soportar los esfuerzos sobre los ejes.

El eje principal se atascaba en la conexión de la base esto no permitía que gire con suficiente fuerza para mover todo el mecanismo lo que se tuvo que hacer fue refrendar el eje en un torno 1.5 mm lo cual dio suficiente espacio para que gire con mayor libertad.

El resto de la construcción fue más sencillo ya que utilizamos tubo PVC solo era de acoplar cada uno de los elemento y atornillarlos para una fijación completa.

Para la construcción de las placas se utilizo el programa PROTEUS con el cual se trazo PCB y se lo traslado a la baquelita mediante una lamina de térmica y se la fundió en acido férrico obteniendo buenos resultados.

Se utilizo acrílico y perfiles de aluminio para la construcción del la caja de control, ya que este material son de bajo costo y de fácil manipulación.

A continuación se mostraran algunas fotos del prototipo de robot manipulador con visión artificial:

Figura 88. Base anclada a la tabla y con la caja de borneras listas

En esta foto observamos que la base la caja de borneras y el brazo un se encuentran conectados.

Figura 89. Materiales en PVC para acoples de los brazos.

Se puede observar los acoples de PVC que se utilizaron en la unir con de los brazos y la base para sostener los ejes, están las uniones de codo que sirvieron para unir el eje 2 que se encuentra girando en el brazo uno, con el brazo 2.

(a)

Figura 90. Brazo completo sin efector final.

Se puede observar el brazo en su posición inicial con sus 2 brazos listps para moverse pero sin el efector final.

(b)

Figura 91. Efector final en proceso de construcción.

(a)

Figura 92. Efector final parte inferior.

En esta foto podemos observar que en la parte final del efector se encuentra el sensor fin de carrera señalado con el numero 1, y el electroimán señalado con el numero 2.

(b)

Figura 93. Caja de control donde se encuentran todos los circuitos.

En esta foto podemos observar la caja e control ya armada y cerrada con los circuitos de control en su interior y conectados a las entrad y salidas en las borneras.

En este capítulo se da a conocer los resultados obtenidos de este trabajo, recalcando la incertidumbre de actuación del Prototipo de robot manipulador con visión artificial desde su inicio.

El control visual tiene la capacidad de proveer una solución de automatización de bajo costo y mantenimiento para ambientes industriales que requieren mayor flexibilidad en sus procesos. El incorporar un sistema de visión por computadora en el lazo de control de un robot manipulador se conoce como control visual.

La mayor parte de las investigaciones en este campo se encuentran en las siguientes categorías:

- Control basado en la posición: En esta la información visual es interpretada con respecto a las coordenadas en el espacio del sistema. Requiere conocer completamente el modelo cinemática del robot, la localización exacta del objetivo o posición deseada y un modelo preciso de calibración para el sistema de visión por computadora. Todos estos requerimientos hace que esta técnica sea muy difícil de implementar en ambientes no estructurados.

- Control basado en la imagen: En esta se usa información obtenida directamente del sistema de visión por computadora (plano de la imagen) como entrada al controlador. Esta técnica es la usada en este trabajo.

En las primeras pruebas que tuvo el robot en modo automático los resultados obtenidos no eran favorables para continuar con mi investigación ya que al capturar las imágenes con las piezas de prueba no ubicaba la posición exacta de estas, sin embargo esto provoco cambiar de posición la cámara ya que también influenciaba la sombra de dicho dispositivo para el reconocimiento de la visión artificial. Los microcontroladores utilizados en el desarrollo del robot manipulador tipo scara con visón artificial, son suficientes para el control, de un prototipo, ya que la principal función es la transferencia de datos.

Como resultados y aplicaciones en la industria de esta labor cabe destacar la consecución de diversos proyectos de investigación y desarrollo dentro del área de Visión Artificial, Robótica y Automatización industrial.

Las ventajas obtenidas en la industria al implementar estos sistemas son considerables, destacándose por ejemplo:

- Mejora en la calidad de los productos y de los procesos involucrados: detección de defectos más pequeños, manipulado más preciso de piezas.

- Mejora en la cantidad de los productos y de los procesos involucrados: mayor rapidez en la inspección de un producto, aumento de la cadencia en la realización de un proceso.

- Integración en el entorno automatizado, dotando de información sensorial en la realización de un proceso automatizado.

- Incremento de la fiabilidad de los procesos, eliminando criterios subjetivos que aparecen en la ejecución de tareas rutinarias, como en los cambios de turno o a lo largo de un turno.

- Manipulado de piezas (pale tizadas, apiladas) y gestión de almacenes (todo tipo de industria).

- Mecanizado de piezas y ajuste de aparatos o máquinas. (Industria básica de transformación de la madera o aluminio).

- Ensamblaje y des ensamblaje automatizado (Industria de empaquetado, del mueble, embalaje).

- Tele operación y apoyo al operador. Empleo de realidad virtual y técnicas de tele operación para facilitar trabajos (Mantenimiento industrial en entornos hostiles).

5.1.1 Interfaz grafica de control PC Robot

Al hacer correr el programa de control desarrollado en Matlab, se obtuvieron los siguientes resultados:

Se abre una ventana con el control principal para controlar el prototipo de visión artificial.

Se definirá la **Figura 94** punto por punto.

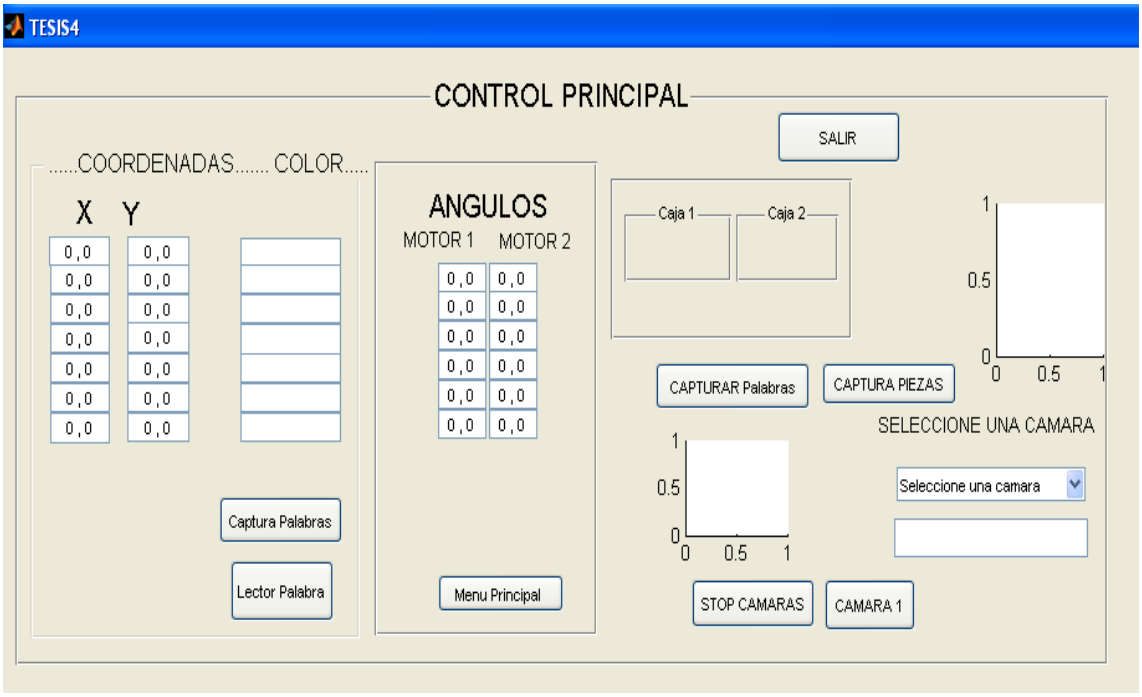


Figura 94. Pantalla de control programada en matlab.

Se puede observar en el lado izquierdo de la pantalla de control las coordenadas rectangulares que se obtendrán al ejecutar el programa.

En el centro del control, se observa los angulos que tiene que mover cada motor para llegar a los puntos donde están las fichas a escoger.

En el lado derecho de la pantalla de control se tiene los botones de control, selección, y de inicio del sistema.

5.1.1.1 Manejo de la pantalla de control

A continuación se enumeraran los pasos a seguir para manejar el software de control del prototipo de robot manipulador con visión artificial.

1. Ejecutamos el sistema
2. Seleccionamos la cámara con la que deseamos trabajar
3. Se activa el botón de inicio
4. Seleccionamos el botón capturar piezas o capturar palabras
5. Damos click en inicio

En este momento se ejecuta el programa y se obtendrán la posición de las cajas y su color además de la posición de cada pieza, como podemos ver en la **Figura 95**.

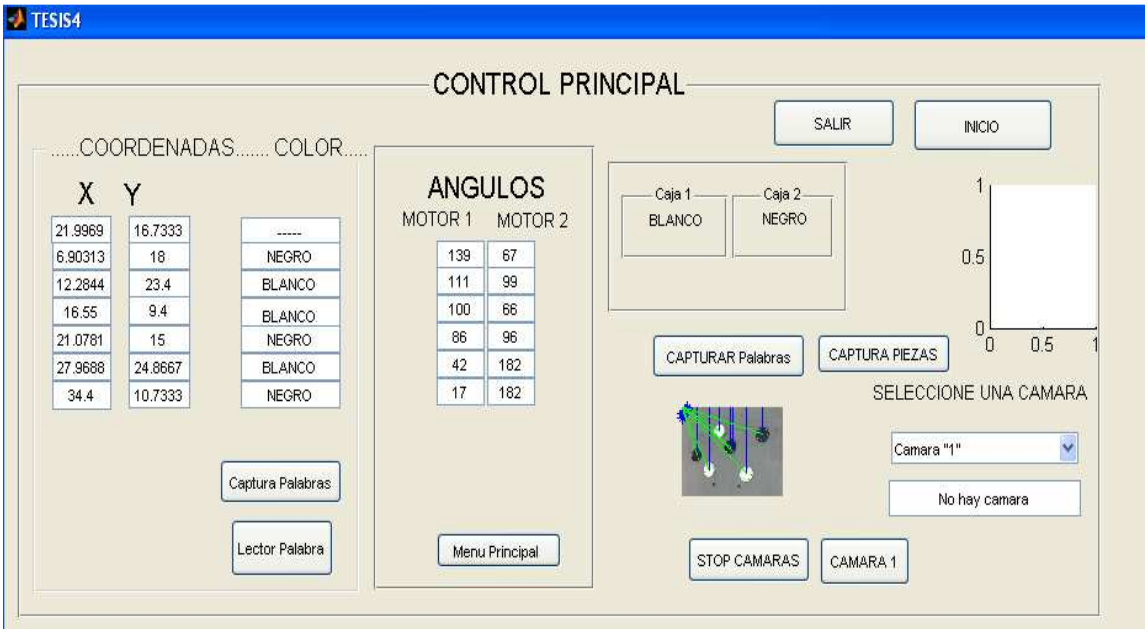


Figura 95. Pantalla de control, después de iniciar.

La información de las coordenadas es enviada atreves del puerto serial. Hacia la tarjeta electrónica de control principal.

5.1.2 Tarjeta electrónica y caja de control.

Los resultados obtenidos con la caja de control y tarjetas electrónicas, son los esperados ya que se realizo el diseño electrónico y se simulo en proteus.

En la **Figura 96** se observa las placas electrónicas de control principal y de control de los servomotores.

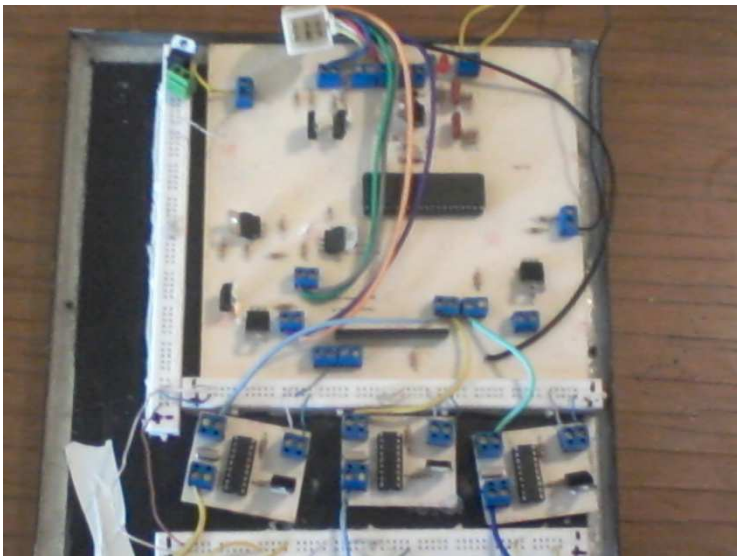


Figura 96. Tarjeta electrónica construida y conectada.

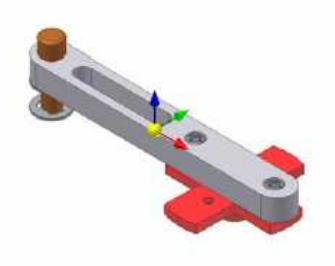
5.2 Rediseño del Efector Final.

Sobre la marcha se tomo una decisión de rediseñar el efector final ya que este no cumplía con los requerimientos necesarios para el correcto funcionamiento del sistema.

Así que se decidí diseñar un sistema de biela manivela el cuan se colocara en sobre el segundo brazo en la parte final de este.

El nuevo diseño de sistema biela manivela se realizo en SOLIDWORKS, obteniendo buenos resultados.

Diseño de Manivela en SOLIDWORKS.



.

Figura 97. Diseño de una manivela para efector final.

Diseño de Biela en SOLIDWORKS.

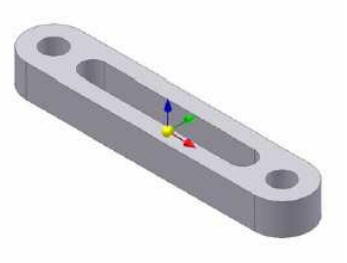


Figura 98. Diseño de una biela para efector final.

Diseño de efector final en SOLIDWORKS.

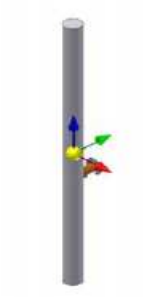


Figura 99. Rediseño, de efector final sobre la marcha.

Diseño de ensamble efector final en SOLIDWORKS.

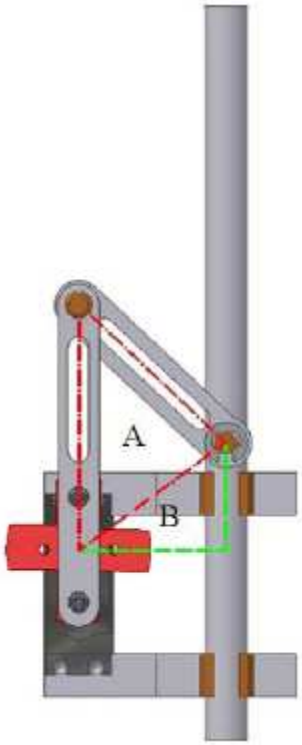


Figura 100. Rediseño, de efector final sobre la marcha.

Realizado el diseño de un nuevo efector final sobre la marcha de construcción del robot manipulador con visión artificial, se comenzó a realizar la construcción, en un torno de las piezas diseñadas.

6. Conclusiones y recomendaciones

6.1. Conclusiones

1) De este trabajo realizado se determina que el proceso de diseño y construcción de un prototipo de robot manipulador tipo scara con visión artificial, es factible.

2) En el desarrollo de la parte mecánica se determinó que es factible la utilización de la materia prima como es el PVC el cual es de bajo costo, resistente, y muy liviano, es perfecto para el desarrollo de la selección de piezas, de un tamaño, peso y temperatura moderado, en un ambiente frio o templado.

3) La utilización de plástico PVC permite reutilizar el material y de esta manera realizar robots con materiales reciclables y de alta durabilidad, contribuyendo con el planeta.

4) En el desarrollo del software de control y visión artificial, se determino que es suficiente para la selección de piezas y de reconocimiento de palabras como en este caso blanco y negro.

5) Para la programación se utilizaron los programas MATLAB Y PIC BACIS PRO, el primero es el que se utilizo en la universidad y esto permitió su buen manejo, con la versión estudiantil, y PBP porque es un excelente copilador de PICs y de muy alto nivel y suficiente para la elaboración de un sistema que transmite datos en forma serial.

6.2. Recomendaciones

1) La recomendación que se puede aplicar en el futuro es en la parte mecánica que se debe incluir algún tipo de rodamiento para que los ejes no tengan tantos esfuerzos. Se debe aumentar la potencia de los 3 servomotores para un mejor movimiento sin trabas en el mecanismo.
2) Para tener mayor estabilidad en la parte mecánica del sistema se recomienda la utilización de rodamientos diseñados para este proceso.
3) El plástico PVC a comparación de aluminio o hierro no tiene la suficiente firmeza para trabajos de mayor fuerza por que se recomiendo la utilización de aluminio en la construcción de un robot que desee controlar mas fuerzas

4) El diseño del efector final podría mejorarse al tener otro modelo para poder tomar piezas que no sean solamente de metal, ya que se utilizo un electroimán en el sistema cuando en realidad se debería utilizar una pina de agarre o una acción por succión para un mejor desempeño.
5) Se debe utilizar una manguera para transportar el cable por el robot, será de mucha utilidad ya que así no se estará en medio de tanto cableado.
6) Al momento de la construcción, seria recomendable realizar una revisión general al diseño. Esto debido a que como se utilizo en la mayoría software computacional para establecer el comportamiento del mismo, es importante respaldar esas simulaciones antes de su construcción.

7) Es fundamental estar familiarizado con el control de MATLAB al momento de utilizar sus herramientas que, al ser correctamente determinadas, resultan ser una gran ayuda.

BIBLIOGRAFÍA

Ponce P. (2010). Inteligencia artificial con aplicación a la ingeniería ($1^a. Ed$). México: Alfaomega.

Boylestad R. (2009). Electrónica y Circuitos eléctricos ($5^a. Ed$). México: Prentice Hall.

Mott R. (2006). Diseño De Elementos De Maquinas ($4^a. Ed$). México: Pearson Educacion.

Baxes G.A. (1994). Digital image processing ($1^a. Ed$). EEUU: John Wiley & Sons.

Vélez Serrano J. (2003). Visión Por Computador. Dykinson, S.L

Brox P. (2010). Fuzzy motion adaptive algorithm for video. Berlin: Springer

Coppin B. (2004). Artificial Intelligence Llluminated. EEUU: Jones&Bartlett.

Esther de Ves Cuenca. (2005). MATLAB: Introducción al procesamiento de imágenes.

Fuentes Covarrubias R. (2002). Inteligencia artificial Teoría y proyectos ($3^a. Ed$). . Colombia:

Lombard, M, (2011). SolidWorks 2011 Assemblies Bible, Estados Unidos: Wiley Publishing Inc.

Moore, H, (2007). Matlab para Ingenieros. Mexico: Pearson Educación.

http://www.matpic.com/

http://mathworks.com

http://campusvirtual.unex.es/cala/epistemowikia/images/1/13

http://www.mecatronicaecuador.com

http://www.um.es/docencia/barzana/IATS/Iats09.html

http://www.um.es/docencia/barzana/IATS

7. ANEXOS

ANEXO1

CODIGO EN PIC BASIC

```
 include "modedefs.bas"
trisa=%11111111
trisc=%11111111
trisb=%11000000
trisE=%0000000
porta=%00000000
portb=%00000000
portc=%00000000
PORTB=0
PORTE=0
ADCON1=%00000000
DATO VAR WORD
DATO1 VAR WORD
envio var word
pm1 var word
pm2 var word
pm3 var word
p var byte

velocidad var byte
x var byte
 y var byte
 z var byte
 N VAR BYTE
 DATO2 VAR WORD
 DDATO VAR BYTE
 DDATO1 VAR BYTE
 define LCD_DREG PORTD
 define LCD_DBIT 4
 define LCD_RSREG PORTD
 define LCD_RSBIT 3
 define LCD_EREG PORTD
 define LCD_EBIT 2
 PAUSE 1000
 LCDOUT $FE,1,"     SISTEMA"
 LCDOUT $FE,$C0,"      SCARA"
 PAUSE 1000
LCDOUT $FE,1,"     MENU:"
PAUSE 1000

 x=50
 Y=90
 z=150
 VELOCIDAD=0
INICIO:

porta=%00000000
portb=%00000000
portc=%00000000

LCDOUT $FE,1,"1) AUTONOMO"
LCDOUT $FE,$C0,"2) CONTROLADO"
PAUSE 50

IF PORTC.0=0 THEN GOTO REBOTE1
IF PORTC.1=0 THEN GOTO REBOTE2

GOTO INICIO

AUTONOMO:
HIGH PORTE.0
LCDOUT $FE,1,"1) DIGITAL"
LCDOUT $FE,$C0,"2) ANALOGO"
PAUSE 50
IF PORTC.0=0 THEN GOTO REBOTE11
IF PORTC.1=0 THEN GOTO REBOTE22
GOTO AUTONOMO
```

```
DIGITAL:

LCDOUT $FE,1,"1) DIGITAL PULSOS"

LCDOUT $FE,$C0,"2) MENU"

PAUSE 50

IF PORTC.0=0 THEN GOTO DIGITALPULSO11

IF PORTC.1=0 THEN GOTO INICIO


GOTO DIGITAL

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'DIGITALPULSO:

'LCDOUT $FE,1,"1) MOTOR1"

'LCDOUT $FE,$C0,"2) MOTOR2"

'PAUSE 5

'IF PORTC.0=0 AND PORTC.6=0 THEN GOTO DIGITALPULSO11

'IF PORTC.1=0 AND PORTC.7=0 THEN GOTO DIGITALPULSO22

'GOTO DIGITALPULSO


''''''''''''''''''''''''''''''''''''''''


DIGITALPULSO11:


pause 20

IF PORTC.0=1 THEN GOTO  DIGITALPULSO1O

GOTO DIGITALPULSO11


'DIGITALPULSO22:

'IF PORTC.1=1 THEN GOTO  DIGITALPULSO2

'GOTO DIGITALPULSO22

''''''''''''''''''''''''''''''''''''''''

DIGITALPULSO1O:

pause 100

LCDOUT $FE,1,"1) 1"

LCDOUT $FE,$C0,"2) 5"

PAUSE 75

IF PORTC.0=0 THEN VELOCIDAD=1

IF PORTC.1=0 THEN VELOCIDAD=5

IF VELOCIDAD>0 THEN GOTO DIGITALPULSO1

goto DIGITALPULSO1O


''''''''''''''''''''''''''''''''''''''''


DIGITALPULSO1:

porte.2= not portb.6


IF PORTC.0=0 THEN GOTO INICIO

porte.2=not portb.6

LCDOUT $FE,1,"1) MENU"

PAUSE 50


SEROUT PORTB.0, N9600,[x] 'Envía x por el pin portb.1 (control)

PAUSE 5

gosub timer 'Subrutina de chequeo del pulsador

goto DIGITALPULSO2

timer:

if portc.2=0 then gosub mas 'Si el p esta a tierra, ir a subr "mas"

if portc.3=0 then gosub menos 'Si el p esta a tierra, ir a subr "menos"

 IF PORTC.0=0 THEN GOTO INICIO

return

mas: 'Subrutina de aumento de posición

pause 5

x=x+velocidAD ; 'Aumento del pulso de salida

if x>=250 then x=250 'Determina un valor máximo de 200

return

menos: 'Subrutina de disminución de posición

pause 5

x=x-VELOCIDAD 'Disminución del pulso de salida

if x=<50 then x=50 'Determina un valor mínimo de 100
```

```
return

GOTO DIGITALPULSO1

''''''''''''''''


DIGITALPULSO2:


porte.2=not portb.6

IF PORTC.0=0 THEN GOTO INICIO

porte.2=not portb.6


LCDOUT $FE,1,"1) MENU"

PAUSE 5

SEROUT PORTB.1, N9600,[y] 'Envía x por el pin
portb.1 (control)

PAUSE 5

gosub timer1 'Subrutina de chequeo del pulsador

GOTO DIGITALPULSO3


timer1:

if portc.4=0 then gosub mas1 'Si el p esta a tierra, ir a
subr "mas"

if portc.5=0 then gosub menos1 'Si el p esta a tierra, ir
a subr "menos"

  IF PORTC.0=0 THEN GOTO INICIO

return

mas1: 'Subrutina de aumento de posición

pause 5

y=y+VELOCIDAD ; 'Aumento del pulso de salida

if y>=250 then y=250'Determina un valor máximo de
200

return

menos1: 'Subrutina de disminución de posición

pause 5

y=y-VELOCIDAD 'Disminución del pulso de salida

if y=<90 then y=90 'Determina un valor mínimo de 100

return

GOTO DIGITALPULSO2


''''''''''''''''

DIGITALPULSO3:

porte.2=not portb.6

IF PORTC.0=0 THEN GOTO INICIO

porte.2=not portb.6


LCDOUT $FE,1,"1) MENU"

PAUSE 5

SEROUT PORTB.2, N9600,[z] 'Envía x por el pin
portb.1 (control)

PAUSE 5

gosub timer2 'Subrutina de chequeo del pulsador

GOTO DIGITALPULSO1


timer2:

if portc.6=0 then gosub mas2 'Si el p esta a tierra, ir a
subr "mas"

if portc.7=0 then gosub menos2 'Si el p esta a tierra, ir
a subr "menos"

  IF PORTC.0=0 THEN GOTO INICIO

return

mas2: 'Subrutina de aumento de posición

pause 5

z=z+VELOCIDAD ; 'Aumento del pulso de salida

if z>=250 then z=250'Determina un valor máximo de
200

return

menos2: 'Subrutina de disminución de posición

pause 5

z=z-VELOCIDAD 'Disminución del pulso de salida

if z=<150 then z=150 'Determina un valor mínimo de
100

return

GOTO DIGITALPULSO3

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

ANALOGO:

porte.2=not portb.6

 IF PORTC.0=0 THEN GOTO INICIO

lcdout $FE,1,"BR1 BR2  MENU"

 lcdout $FE,$C0,dec3 dato1,"  ",dec3 Ddato1," ","1)"
```

```
 pause 50

Adcin      0,dato

ADCIN   1,DDATO

dato1=dato*151/255+60  ' calculo promedio para control de servo motor

DDATO1=DDATO*151/255+60

 IF PORTC.0=0 THEN GOTO INICIO

 lcdout $FE,1,"BR1 BR2  MENU"

 lcdout $FE,$C0,dec3 dato1,"  ",dec3 Ddato1 ," ","1)"


 SEROUT PORTB.0, N9600,[DATO1] 'Envía x por el pin portb.1 (control)

PAUSE 5

  SEROUT PORTB.1, N9600,[DDATO1] 'Envía x por el pin portb.1 (control)

PAUSE 5

GOTO ANALOGO


''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

REBOTE1:

IF PORTC.0=1 THEN GOTO AUTONOMO

GOTO REBOTE1


REBOTE2:

IF PORTC.1=1 THEN GOTO CONTROLADO

GOTO REBOTE2


REBOTE11:

IF PORTC.0=1 THEN GOTO DIGITAL

GOTO REBOTE11


REBOTE22:

IF PORTC.1=1 THEN GOTO ANALOGO

GOTO REBOTE22


''''''''''''''''''''''''''''''''''''''''''''''''''''''''

''''''''''''''''''''''''''''''''''''''''''''''''''''''''

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

CONTROLADO:

  SEROUT PORTB.5, N9600,["E"] 'Envía x por el pin portb.1 (control)

PAUSE 150

lcdout $FE,1,"Pulsar Salir"

 pause 5

 SERIN PORTB.7,N9600, envio

 controlado1:

IF envio=="N" THEN goto pos1

if envio =="B" then goto pos2

  goto controlado1

   pos1:

  pm1= 150

  pm2= 90

  pm3= 150

  SEROUT PORTB.0, N9600,[pm1] 'Envía x por el pin portb.1 (control)

PAUSE 150

  SEROUT PORTB.1, N9600,[pm2] 'Envía x por el pin portb.1 (control)

PAUSE 150

 SEROUT PORTB.2, N9600,[pm3]

pause 150

 goto mover

  pos2:


 pm1= 50

 pm2= 90

 pm3= 150


  SEROUT PORTB.0, N9600,[pm1] 'Envía x por el pin portb.1 (control)

PAUSE 150

  SEROUT PORTB.1, N9600,[pm2] 'Envía x por el pin portb.1 (control)

PAUSE 150

 SEROUT PORTB.2, N9600,[pm3]
```

```
pause 150

goto mover

mover:

SERIN PORTB.7,N9600, pm1

pause 5

SERIN PORTB.7,N9600, pm2

pause 5

SEROUT PORTB.0, N9600,[pm1] 'Envía x por el pin
portb.1 (control)

PAUSE 150

  SEROUT PORTB.1, N9600,[pm2] 'Envía x por el pin
portb.1 (control)

PAUSE 150

efector:

for  p=1 to 50

pm3=pm3+5

 SEROUT PORTB.2, N9600,[pm3]

pause 150

if portb.6=0 then high porte.2

if portb.6=0 then goto efectorr

next

efectorr:

for  p=1 to 50

pm3=pm3-25

 SEROUT PORTB.2, N9600,[pm3]

pause 150

if portb.6=0 then goto efectorr

next


regreso:

if envio="N" then goto pos11

if envio="B" then goto pos22

goto regreso

 pos11:

  pm1= 150

  pm2= 90

  pm3= 150

SEROUT PORTB.0, N9600,[pm1] 'Envía x por el pin
portb.1 (control)

PAUSE 150

  SEROUT PORTB.1, N9600,[pm2] 'Envía x por el pin
portb.1 (control)

PAUSE 150

 SEROUT PORTB.2, N9600,[pm3]

pause 150

 goto efector1

  pos22:

  pm1= 50

 pm2= 90

 pm3= 150

    SEROUT PORTB.0, N9600,[pm1] 'Envía x por el pin
portb.1 (control)

PAUSE 150

  SEROUT PORTB.1, N9600,[pm2] 'Envía x por el pin
portb.1 (control)

PAUSE 150

 SEROUT PORTB.2, N9600,[pm3]

pause 150


goto efector1

efector1:

for  p=1 to 50

pm3=pm3+5

 SEROUT PORTB.2, N9600,[pm3]

pause 150

if portb.6=0 then low porte.2

if portb.6=0 then goto efectorr1

next

efectorr1:

for  p=1 to 50

pm3=pm3-25

 SEROUT PORTB.2, N9600,[pm3]

pause 150

if portb.6=0 then goto efectorr

next

goto CONTROLADO
```

ANEXO2

CODIGO EN MATLAB

```
function varargout = TESIS4(varargin)
% TESIS4 M-file for TESIS4.fig
%      TESIS4, by itself, creates a new TESIS4 or raises the
existing
%      singleton*.
%
%      H = TESIS4 returns the handle to a new TESIS4 or the
handle to
%      the existing singleton*.
%
%      TESIS4('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in TESIS4.M with the given
input arguments.
%
%      TESIS4('Property','Value',...) creates a new TESIS4 or
raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before TESIS4_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes
property application
%      stop.  All inputs are passed to TESIS4_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose
"GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help TESIS4

% Last Modified by GUIDE v2.5 21-Apr-2012 22:57:04

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @TESIS4_OpeningFcn, ...
                   'gui_OutputFcn',  @TESIS4_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before TESIS4 is made visible.
function TESIS4_OpeningFcn(hObject, eventdata, handles,
varargin)
handles.SerPIC = serial('COM1');%Seleccionar puerto COM1
set(handles.SerPIC,'BaudRate',9600);%Velocidad 2400
baudios
set(handles.SerPIC,'DataBits',8);%8 bits de datos
set(handles.SerPIC,'Parity','none');%Sin control de paridad
set(handles.SerPIC,'StopBits',1);%Un bit de parada
set(handles.SerPIC,'FlowControl','none');%Sin control de flujo
%fopen(handles.SerPIC);            %--Abrir el puerto serial
fopen(handles.SerPIC);            %--Abrir el puerto serial
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes TESIS4 wait for user response (see
UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command
line.
function varargout = TESIS4_OutputFcn(hObject, eventdata,
handles)

varargout{1} = handles.output;


% --- Executes on button press in INICIO.
function INICIO_Callback(hObject, eventdata, handles)
global Ne;
%_____________________________________________
_____
%RESETEAR VALORES
set(handles.cox1,'string',['']);
set(handles.cox2,'string',['']);
set(handles.cox3,'string',['']);
set(handles.cox4,'string',['']);
set(handles.cox5,'string',['']);set(handles.ccc1,'string',['']);
set(handles.cox6,'string',['']);set(handles.ccc2,'string',['']);
set(handles.cox7,'string',['']);set(handles.ccc3,'string',['']);
set(handles.ccc4,'string',['']);
set(handles.ccc5,'string',['']);
set(handles.ccc6,'string',['']);
set(handles.ccc7,'string',['']);


set(handles.cy1,'string',['']);
set(handles.cy2,'string',['']);
set(handles.cy3,'string',['']);
set(handles.cy4,'string',['']);
set(handles.cy5,'string',['']);
set(handles.cy6,'string',['']);
set(handles.cy7,'string',['']);

set(handles.box1,'string',[' ']);
set(handles.box2,'string',[' ']);
%_____________________________________________
_____
%programa de leer BLANKO O NEGRO de las etiketas

imagen=imread('c:\imagen.jpg');
% abrimos la imagen
%imshow(imagen);
caja1='a'
caja2='b'
sn1='N'
sn2='E'
sn3='G'
sn4='R'
%sn5='O'
sb1='B'
sb2='L'
sb3='A'
sb4='N'
sb5='C'
%sb6='O'

sn21='N'
sn22='E'
sn23='G'
sn24='R'
%sn25='O'
sb21='B'
sb22='L'
sb23='A'
sb24='N'
sb25='C'
%sb26='O'


title('')
% PASAMOS LA IMAGEN A ESCALA DE GRISES
imagenI= rgb2gray(imagen); % pasamos a escala de grises la
imagen
imagenI=imagen;
%imshow(imagen)%mostramos la imagen en grises
%pause(0.1)

imagen = imadjust(imagen,[0.49 0.6],[])%contraste de la
imagen en grises, bajamos el contraste para poder filtral y
seleccionar las lineas que necesitamos

%imshow(imagen)
```

```
%pause(0.1)


f = imfill(imagen,'holes');%filtramos la foto, eliminamos lo de
adentro de cada cuadrdado
%imshow(f)
%pause(0.1)

imagenumbral= graythresh(f);%lo mismo de arriba

imagen= im2bw(f,imagenumbral);%convertimos la imagen a
binaria con base en el umbral
%imshow(imagen)% mostramos la imgane en un plot
%pause (0.1)

imagen = imfill(imagen,'holes');%filtramos la foto, eliminamos
lo de adentro de cada cuadrdado
%imshow(imagen)
%pause(0.1)


for n=1 : 3
[L Me]=bwlabel(imagen);%bwlabel etiquetacion de elementos
conectados en la imagen nos retorna en L y numerode
elementos en Ne

propiedades= regionprops(L,'basic'); %usasr la propiedad
region props en la matris de lemenetos conectados con basic
por que solo es un argumento de netrada L
  % calculo basicos de las propidades areas centros y
numeros de cajas para encerrar el onjeto
  %propiedades es una estructura se la puede ver en el
worksapce
 % hold on % conservamos la imagen actual



%%% BUSCAR TIPOS DE AREAS

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
aqui pegue

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
del 1
   buscar=find([propiedades.Area]<3000);%vector de
identificadores
  %encontradas las areas las ponemos en un rectangulo
diferente las areas
  %menores a 800 seran rojas

  %eleiminar areas menores de 800
  for n=1:size(buscar,2)
      d=
round(propiedades(buscar(n)).BoundingBox);%redondeo de
los valor de los rectangulos para la eliminacion
      imagen(d(2):d(2)+d(4),d(1):d(1)+d(3))=0;%los redondeos
de ares entre filas y columnas se hacen 0 de las variables
buscar
  end

  %imshow(imagen)%mostramos sololas figuras con area
mayor a 800
  %pause (0.2)

end
%--------------------------------------------------------------------------

% OJO para la primera etiketa PROGRAMA DE SABER SI ES
NEGRO O BLANCO LA CAJA
 %hold off


imagenS = imcrop(imagenI,propiedades(1).BoundingBox);

%imshow(imagenS)
%pause(0.5)

%hold on

imagenS= rgb2gray(imagenS); % pasamos a escala de grises
la imagen

%imshow(imagenS)%mostramos la imagen en grises
%pause(0.5)

imagenS = imadjust(imagenS,[0.3 0.6],[])%contraste de la
imagen en grises, bajamos el contraste para poder filtral y
seleccionar las lineas que necesitamos
%imshow(imagenS)%mostramos la imagen en grises
%pause(5)

threshold = graythresh(imagenS);
imagenS =~im2bw(imagenS,threshold);
% Remove all object containing fewer than 30 pixels
imagenS = bwareaopen(imagenS,30);
%imshow(imagenS)%mostramos la imagen en grises
%pause(0.5)


%matris para almacenar la palabra
word=[ ];
re=imagenS;
% LLAMAMOS A LA VARIABLE Sletras
load Sletras
global Sletras

% Calculamos el numero de letras en la matris de letras
num_letras=size(Sletras,2);

%_____________________________________________________
_____________________
   %llamamos a la funcion lineas
   [fl re]=lineas(re);
   imgn=fl;  %sacamos la fila 1
   %imshow(fl);
   %pause(1.5) %mostramos la primera linea de la foto

   %-----------------------------------------------------------------
   % ETIKETAMOS LOS COMPONENTES QUE SE
ENCUENTRAN JUNTOS

   [L Ne] = bwlabel(imgn);
   for n=1:Ne
      [r,c] = find(L==n);
      % SACAMOS LAS LETRAS DE CADA linea
      n1=imgn(min(r):max(r),min(c):max(c));
      % CAMBIAMOS EL TAMAÑO DE CADA LETRA
      img_r=imresize(n1,[42 24]);

       %imshow(img_r);pause(0.5)%PARA VER COMO
CAMBIA EL TAMAÑO
      %-----------------------------------------------------------------
      % Call fcn to convert image to text
      letral=leerl(img_r,num_letras);
      % Letter concatenation
      word=[word letral];
      word1=[word]
      worda=word1
   end

%--------------------------------------------------------------------------
%COMPARACION DE LETRAS
%comparacion de palabra1

letraN1=1
for n=1:5
if worda(n)=='N'
sn1='N'
break
else
   sn1=' '
end
letraN1=letraN1+1
end

letraN2=1
for n=letraN1 :Ne
if worda(n)=='E'
sn2='E'
break
else
   sn2=' '
end
letraN2=letraN2+1
end

letraN3=1
for n=letraN2 :Ne
if worda(n)=='G'
sn3='G'
```

```
break
else
   sn3=' '
end
letraN3=letraN3+1
end

letraN4=1
for n=letraN3 :Ne
if worda(n)=='R'
sn4='R'
break
else
   sn4=' '
end
letraN4=letraN4+1
end


%.....

letraB1=1
for n=1:5
if worda(n)=='B'
sb1='B'
break
else
   sb1=' '
end
letraB1=letraB1+1
end

letraB2=1
for n=letraB1 :Ne
if worda(n)=='L'
sb2='L'
break
else
   sb2=' '
end
letraB2=letraB2+1
end

letraB3=1
for n=letraB2 :Ne
if worda(n)=='A'
sb3='A'
break
else
   sb3=' '
end
letraB3=letraB3+1
end

letraB4=1
for n=letraB3 :Ne
if worda(n)=='N'
sb4='N'
break
else
   sb4=' '
end
letraB4=letraB4+1
end

letraB5=1
for n=letraB4 :Ne
if worda(n)=='C'
sb5='C'
break
else
   sb5=' '
end
letraB5=letraB5+1
end

palabra1BF=[sb1,sb2,sb3,sb4,sb5];
palabra1NF=[sn1,sn2,sn3,sn4];

%
%-------------------------------------------------------------------
%-------------------------------------------------------------------
% OJO para la primera etiketa PROGRAMA DE SABER SI ES
NEGRO O BLANCO LA CAJA
% segunda parte, para la segunda parte de laimagen
 %hold off

imagenS = imcrop(imagenI,propiedades(2).BoundingBox);

%imshow(imagenS)
%pause(0.5)

%hold on

imagenS= rgb2gray(imagenS); % pasamos a escala de grises
la imagen

%imshow(imagenS)%mostramos la imagen en grises
%pause(0.5)

imagenS = imadjust(imagenS,[0.3 0.6],[])%contraste de la
imagen en grises, bajamos el contraste para poder filtral y
seleccionar las lineas que necesitamos
%imshow(imagenS)%mostramos la imagen en grises
%pause(5)

threshold = graythresh(imagenS);
imagenS =~im2bw(imagenS,threshold);
% Remove all object containing fewer than 30 pixels
imagenS = bwareaopen(imagenS,30);
%imshow(imagenS)%mostramos la imagen en grises
%pause(0.5)
%matris para almacenar la palabra
word=[ ];
re=imagenS;
% LLAMAMOS A LA VARIABLE Sletras
load Sletras


% Calculamos el numero de letras en la matris de letras
num_letras=size(Sletras,2);

%_____________________________________________________
_____________________
   %llamamos a la funcion lineas
   [fl re]=lineas(re);
   imgn=fl;  %sacamos la fila 1
   %imshow(fl);pause(1.5) %mostramos la primera linea de la
foto

   %-----------------------------------------------------------
   % ETIKETAMOS LOS COMPONENTES QUE SE
ENCUENTRAN JUNTOS

   [L Ne] = bwlabel(imgn);
   for n=1:Ne
      [r,c] = find(L==n);
      % SACAMOS LAS LETRAS DE CADA linea
      n1=imgn(min(r):max(r),min(c):max(c));
      % CAMBIAMOS EL TAMAÑO DE CADA LETRA
      img_r=imresize(n1,[42 24]);

       %imshow(img_r);pause(0.5)%PARA VER COMO
CAMBIA EL TAMAÑO
      %-----------------------------------------------------------
      % Call fcn to convert image to text
      letraI=leerl(img_r,num_letras);
      % Letter concatenation
      word=[word letraI];
      word1=[word]
   end

%-------------------------------------------------------------------
%COMPARACION DE LETRAS
%comparacion de palabra1



letraN1=1
for n=1:(Ne-(Ne-(Ne/2)))
if word1(n)=='N'
sn21='N'
break
else
   sn21=' '
end
letraN1=letraN1+1
end

letraN2=1
for n=letraN1 :Ne
```

```
if word1(n)=='E'
sn22='E'
break
else
   sn22=' '
end
letraN2=letraN2+1
end

letraN3=1
for n=letraN2 :Ne
if word1(n)=='G'
sn23='G'
break
else
   sn23=' '
end
letraN3=letraN3+1
end

letraN4=1
for n=letraN3 :Ne
if word1(n)=='R'
sn24='R'
break
else
   sn24=' '
end
letraN4=letraN4+1
end


%.....

letraB1=1
for n=1:(Ne-(Ne-(Ne/2)))
if word1(n)=='B'
sb21='B'
break
else
   sb21=' '
end
letraB1=letraB1+1
end

letraB2=1
for n=letraB1 :Ne
if word1(n)=='L'
sb22='L'
break
else
   sb22=' '
end
letraB2=letraB2+1
end

letraB3=1
for n=letraB2 :Ne
if word1(n)=='A'
sb23='A'
break
else
   sb23=' '
end
letraB3=letraB3+1
end

letraB4=1
for n=letraB3 :Ne
if word1(n)=='N'
sb24='N'
break
else
   sb24=' '
end
letraB4=letraB4+1
end

letraB5=1
for n=letraB4 :Ne
if word1(n)=='C'
sb25='C'
break
else
   sb25=' '
end
letraB5=letraB5+1
end


cfn1=0;%ojo variable de conteo final
cfb1=0;
cfn2=0;
cfb2=0;
palabra2BF=[sb21,sb22,sb23,sb24,sb25];
palabra2NF=[sn21,sn22,sn23,sn24];

if sn1=='N'
   cfn1=cfn1+1;
  end
if sn2=='E'
   cfn1=cfn1+1;
end
if sn3=='G'
  cfn1=cfn1+1;
end
if sn4=='R'
   cfn1=cfn1+1;
end


if sb1=='B'
   cfb1=cfb1+1;
end
if sb2=='L'
   cfb1=cfb1+1;
end
if sb3=='A'
  cfb1=cfb1+1;
end
if sb4=='N'
   cfb1=cfb1+1;
end
if sb5=='C'
   cfb1=cfb1+1;
end

%.....
if sn21=='N'
   cfn2=cfn2+1;
end
if sn22=='E'
   cfn2=cfn2+1;
end
if sn23=='G'
   cfn2=cfn2+1;
end
if sn24=='R'
   cfn2=cfn2+1;
end


if sb21=='B'
   cfb2=cfb2+1;
end
if sb22=='L'
   cfb2=cfb2+1;
end
if sb23=='A'
    cfb2=cfb2+1;
end
if sb24=='N'
    cfb2=cfb2+1;
end
if sb25=='C'
    cfb2=cfb2+1;
end

%.....


%ojo cambio ultimo
palabra2BF=[sb21,sb22,sb23,sb24,sb25];
palabra2NF=[sn21,sn22,sn23,sn24];

if sn1=='N'&& cfn1>cfb1
   caja1='NEGRO'
end
if sn2=='E'&& cfn1>cfb1
   caja1='NEGRO'
end
```

```
if sn3=='G'&& cfn1>cfb1
    caja1='NEGRO'
end
if sn4=='R'&& cfn1>cfb1
    caja1='NEGRO'
end


if sb1=='B'&& cfb1>cfn1
    caja1='BLANCO'
end
if sb2=='L'&& cfb1>cfn1
    caja1='BLANCO'
end
if sb3=='A'&& cfb1>cfn1
    caja1='BLANCO'
end
if sb4=='N'&& cfb1>cfn1
    caja1='BLANCO'
end
if sb5=='C'&& cfb1>cfn1
    caja1='BLANCO'
end

%.....................................................................
if sn21=='N'&& cfn2>cfb2
    caja2='NEGRO'
end
if sn22=='E'&& cfn2>cfb2
    caja2='NEGRO'
end
if sn23=='G'&& cfn2>cfb2
    caja2='NEGRO'
end
if sn24=='R'&& cfn2>cfb2
    caja2='NEGRO'
end


if sb21=='B'&& cfb2>cfn2
    caja2='BLANCO'
end
if sb22=='L'&& cfb2>cfn2
    caja2='BLANCO'
end
if sb23=='A'&& cfb2>cfn2
    caja2='BLANCO'
end
if sb24=='N'&& cfb2>cfn2
    caja2='BLANCO'
end
if sb25=='C'&& cfb2>cfn2
    caja2='BLANCO'
end

pause(0.5)
%close all

%_____________________________________________________
_____________________
%COLOR DE LAS CAJAS
set(handles.box1,'string',[caja1]);
set(handles.box2,'string',[caja2]);
%_____________________________________________________
_____________________



%programa principal

%tomar una imagen para seleccionar el nuevo puntero
%uiwait(msgbox('Capturar piezas','Atencion','modal'));
%pause(1)
%imagenprincipal = getsnapshot(handles.objVideo);
%imwrite(imagenprincipal, 'c:\i1.jpg') ;

imagen=imread('c:\i1.jpg');% leemos la imagen que esta
guarda en el disco c de extencion jpg

handles.imagen=imread('c:\i1.jpg');%ACTIVAMMOS EN EL
OBJETO VIDEO LA CAMARA 1
guidata(hObject,handles);

axes(handles.axes2)
imshow(imagen)%mostramos la imagen normal
pause(0.01)

imagen=imread('c:\i1.jpg');% leemos la imagen que esta
guarda en el disco c de extencion jpg
imagengris=  rgb2gray (imagen); % pasamos a escala de
grises la imagen
imagenedge= edge (imagengris, 'prewitt');

SE= strel('square',8)%...

imagendila= imdilate(imagenedge,SE)% DILATAMOS LAS
LINEAS c
imagen1= not (imagendila)% invertimos los colores de blanco
a negro


for n=1 : 3
[L Ne]=bwlabel(imagen1);%bwlabel etiquetacion de elementos
conectados en la imagen nos retorna en L y numerode
elementos en Ne

propiedades= regionprops(L,'basic'); %usasr la propiedad
region props en la matris de lemenetos conectados con basic
por que solo es un argumento de netrada L
  % calculo basicos de las propidades areas centros y
numeros de cajas para encerrar el onjeto
  %propiedades es una estructura se la puede ver en el
worksapce



%%% BUSCAR TIPOS DE AREAS

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
aqui pegue

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
del 1
   buscar=find([propiedades.Area]<100);%vector de
identificadores
  %encontradas las areas las ponemos en un rectangulo
diferente las areas
  %menores a 800 seran rojas



  %eleiminar areas menores de 800
  for n=1:size(buscar,2)
      d=
round(propiedades(buscar(n)).BoundingBox);%redondeo de
los valor de los rectangulos para la eliminacion
      imagen1(d(2):d(2)+d(4),d(1):d(1)+d(3))=0;%los redondeos
de ares entre filas y columnas se hacen 0 de las variables
buscar
  end


  imagen1= not(imagen1)

end



  for n=1:size(propiedades,1)% no lo usare por que cadad
cuadra tendra difernte color

      x=propiedades(n).Centroid(1);%buscamos el centro de
cada figura con la propieda centroid
     y=propiedades(n).Centroid(2);
      end
  %-----------------------------------------------------------
a=0;
b=0;
nu=0;
PIEZA={''};

aa=0.0;
bb=0.0;

imagen2=imagengris<=128


   for n=1:size(propiedades,1)% no lo usare por que cadad
cuadra tendra difernte color
```

```
      x=propiedades(n).Centroid(1);%buscamos el centro de
cada figura con la propieda centroid
     y=propiedades(n).Centroid(2);

 xr= round(x);% REDONDIAMOS LOS PUNTOS DE LOS
EJES PARA PODER TYRABAJAR CON BINARIOS
 yr= round(y);

d(n)=imagen2(yr, xr)%obtenemos el valor de esa cordenada
para ver si es cero o uno negro o blanco
dxr(n)=xr
dyr(n)=yr
   end
%------------------------------------------------- en esta pate
sabemos
%que pieza es blanca y cual es negra y la nombramos a su
variable
for N=1:Ne
if d(N)==0
nu={'BLANCO'};
else
nu={'NEGRO'};
end
PIEZA(N)=nu;
end
PIEZAS=char(PIEZA)
%_____________________________________________________
_____________-____

 %transformacion a medidas los pixeles
 medidaIG=[size(imagengris)]% medida de la imagen gris

 medidaIGY=medidaIG(1)%medida de la y de los pixeles de la
imagen gris
 medidaIGX=medidaIG(2)%medida de la x de los pixeles de la
imagen gris

 medidaTTY=32;%medida de  la tabla de trabajo y
 medidaTTX=42%mediad de la tabla de trabajo x


for n=1 :Ne
 dxrR(n)=((dxr(n)*medidaTTX)/ medidaIGX); %calculamos los
pixeles por centimetros de la tabla de trabajo x
 dyrR(n)=((dyr(n)*medidaTTY)/ medidaIGY);%calculamos los
pixeles por centimetros de la tabla de trabajo y
end

 for n=1 :Ne
 dxrR(n)=(dxrR(n)+0.8); %calculamos los pixeles por
centimetros de la tabla de trabajo x
 dyrR(n)=(dyrR(n)+0.8);%calculamos los pixeles por
centimetros de la tabla de trabajo y
end

 %-------------------------------------------------------------------------

 %PARTE DEL ANGULOS Y BRAZOS
 d1=15
d2=20
q1=90
q2=90
%ecuaciones para el area de trabajo del brazo
hold on

for q1= 0 : 180
   for q2=0 :8
x1=(d1*sind(q1)+d2*sind(q1+q2))
y1=(d1*cosd(q1)+d2*cosd(q1+q2))

x2=[0 x1]
y2=[60 y1]
plot(x2,y2,'*')

  end
end


pause(0.1)



for n=1: Ne
%y o cateto del punto de la cordenada
c1x=[0 dxr(n) dxr(n)]
c1y=[0 dyr(n) 0]
plot(c1x,c1y,'b')

%x o cateto del punto de la cordenada
c2x=[dxr(n) 0]
c2y=[0 0]
plot(c2x,c2y)

%trazamos la hipotenusa entre los lados formados por la
coordenada
h1x=[0 dxr(n)]
h1y=[0 dyr(n)]
plot(h1x,h1y,'-g')

%hipotenusa
h=sqrt((dxrR(n)^2)+(dyrR(n)^2));%%OJO R r

%angulo alfa del triangulo entre coordenadas
 alfa=asind(dyrR(n)/h);%cateto opusteo siempre y en este
sistema %%OJO R
%angulo beta entre la hipotenusa y el brazo
%beta=asind(d2/d1)
beta=acosd((d1^2-d2^2+h^2)/(d1*2*h));
%angulo del brazo

angulobrazo=(alfa+beta);

%delta es el angulo opuesto de la hipotenusa encotrada
anteriormente que
%paso hacer cateto dividido para d1 que es el lado mas
grande

%delta=asind(h/d1)
delta=acosd((d1^2+d2^2-h^2)/(d1*2*d2))+1.5;
pause(0.8);


angulobrazor=round (angulobrazo);
deltar=round(delta);

dangulobrazor(n)=angulobrazor;
ddeltar(n)=deltar;

end

for ccc=1:Ne
 pul1(ccc)=dangulobrazor(ccc)*150/90
 pulso1(ccc)=round(pul1(ccc))

 pul2(ccc)=ddeltar(ccc)*150/90
 pulso2(ccc)=round(pul2(ccc))
end

%_____________________________________________________
_____________________


%_____________________________________________________
_____________________
  %ponemos las cordenadas en X y Y

%_____________________________________________________
_____________-
  %poner en los edit box si es blanko o negro
  Nee=Ne-1;
  sss=0;
  if Nee<0
     Nee=0;
  end

  if Nee>0

        if sss<=Ne
           sss=sss+1;
        end

       if PIEZAS(sss)=='B'
          set(handles.ccc1,'string',['-----
']);set(handles.cox1,'string',[dxrR(sss)]);set(handles.cy1,'string'
,[dyrR(sss)]);

       elseif PIEZAS(sss)=='N'
          set(handles.ccc1,'string',['-----
']);set(handles.cox1,'string',[dxrR(sss)]);set(handles.cy1,'string'
,[dyrR(sss)]);
```

```
        end

        if sss<=Ne
            sss=sss+1;
        end


        if PIEZAS(sss)=='B'

set(handles.ccc2,'string',['BLANCO']);set(handles.cox2,'string',[
dxrR(sss)]);set(handles.cy2,'string',[dyrR(sss)]);
set(handles.mt1,'string',[dangulobrazor(sss)]);set(handles.mt2
1,'string',[ddeltar(sss)]);

        elseif PIEZAS(sss)=='N'

set(handles.ccc2,'string',['NEGRO']);set(handles.cox2,'string',[
dxrR(sss)]);set(handles.cy2,'string',[dyrR(sss)]);
set(handles.mt1,'string',[dangulobrazor(sss)]);set(handles.mt2
1,'string',[ddeltar(sss)]);
        end


        if sss<=Ne
            sss=sss+1;
        end
        if PIEZAS(sss)=='B'

set(handles.ccc3,'string',['BLANCO']);set(handles.cox3,'string',[
dxrR(sss)]);set(handles.cy3,'string',[dyrR(sss)]);
set(handles.mt2,'string',[dangulobrazor(sss)]);set(handles.mt2
2,'string',[ddeltar(sss)]);

        elseif PIEZAS(sss)=='N'

set(handles.ccc3,'string',['NEGRO']);set(handles.cox3,'string',[
dxrR(sss)]);set(handles.cy3,'string',[dyrR(sss)]);
set(handles.mt2,'string',[dangulobrazor(sss)]);set(handles.mt2
2,'string',[ddeltar(sss)]);
        end

        if sss<=Ne
            sss=sss+1;
        end
        if PIEZAS(sss)=='B'

set(handles.ccc4,'string',['BLANCO']);set(handles.cox4,'string',[
dxrR(sss)]);set(handles.cy4,'string',[dyrR(sss)]);
set(handles.mt3,'string',[dangulobrazor(sss)]);set(handles.mt2
3,'string',[ddeltar(sss)]);

        elseif PIEZAS(sss)=='N'

set(handles.ccc4,'string',['NEGRO']);set(handles.cox4,'string',[
dxrR(sss)]);set(handles.cy4,'string',[dyrR(sss)]);
set(handles.mt3,'string',[dangulobrazor(sss)]);set(handles.mt2
3,'string',[ddeltar(sss)]);
        end

        if sss<=Ne
            sss=sss+1;
        end
        if PIEZAS(sss)=='B'

set(handles.ccc5,'string',['BLANCO']);set(handles.cox5,'string',[
dxrR(sss)]);set(handles.cy5,'string',[dyrR(sss)]);
set(handles.mt4,'string',[dangulobrazor(sss)]);set(handles.mt2
4,'string',[ddeltar(sss)]);

        elseif PIEZAS(sss)=='N'

set(handles.ccc5,'string',['NEGRO']);set(handles.cox5,'string',[
dxrR(sss)]);set(handles.cy5,'string',[dyrR(sss)]);
set(handles.mt4,'string',[dangulobrazor(sss)]);set(handles.mt2
4,'string',[ddeltar(sss)]);
        end


        if sss<=Ne
            sss=sss+1;
        end
        if PIEZAS(sss)=='B'

set(handles.ccc6,'string',['BLANCO']);set(handles.cox6,'string',[
dxrR(sss)]);set(handles.cy6,'string',[dyrR(sss)]);
set(handles.mt5,'string',[dangulobrazor(sss)]);set(handles.mt2
5,'string',[ddeltar(sss)]);

        elseif PIEZAS(sss)=='N'

set(handles.ccc6,'string',['NEGRO']);set(handles.cox6,'string',[
dxrR(sss)]);set(handles.cy6,'string',[dyrR(sss)]);
set(handles.mt5,'string',[dangulobrazor(sss)]);set(handles.mt2
5,'string',[ddeltar(sss)]);
        end

        if sss<=Ne
            sss=sss+1;
        end
        if PIEZAS(sss)=='B'

set(handles.ccc7,'string',['BLANCO']);set(handles.cox7,'string',[
dxrR(sss)]);set(handles.cy7,'string',[dyrR(sss)]);
set(handles.mt6,'string',[dangulobrazor(sss)]);set(handles.mt2
6,'string',[ddeltar(sss)]);

        elseif PIEZAS(sss)=='N'

set(handles.ccc7,'string',['NEGRO']);set(handles.cox7,'string',[
dxrR(sss)]);set(handles.cy7,'string',[dyrR(sss)]);
set(handles.mt6,'string',[dangulobrazor(sss)]);set(handles.mt2
6,'string',[ddeltar(sss)]);
        end



    end




%_____
_____

%enviamos la señal al pic con los valores de pulsos que debe
enviar a los
%servos motores
pos1=[0 168 240 70 127 78 168]
pos2=[0 69 74 200 150 127 69]

for recoge=2:Ne

fprintf(handles.SerPIC,'%s',pos1(recoge));

pause(1)

fprintf(handles.SerPIC,'%s',pos2(recoge));
pause(1)
    if PIEZAS(recoge)=='N'
        fprintf(handles.SerPIC,'%s',125);
        pause(0.3)
        fprintf(handles.SerPIC,'%s',105);%ojo
    end

    if PIEZAS(recoge)=='B'
        fprintf(handles.SerPIC,'%s',225);
        pause(0.3)
        fprintf(handles.SerPIC,'%s',225);%ojo
    end
pause(8)
 end
%_____
_____
%SELECCION DE CAMARAS DE UNA LISTA

% --- Executes on selection change in camaras.
function camaras_Callback(hObject, eventdata, handles)

ccamaras=IMAQHWINFO('winvideo')

ss=0;
cc2=ccamaras.DeviceIDs%encontramos un arregloc el de las
camaras conectadas

cc3=cell2mat(cc2)% transformamos de arreglo cell a mat
nc=1%nuemr de camaras

for nnc=1:cc3
    ss=1+ss
end
```

```
if isempty(cc3)
    cam1='No hay camara';
    cam2='No hay camara';
    cam3='No hay camara';
    ss=0;
end




if ss==1
    cam1=IMAQHWINFO('winvideo',1)
    cam2='No  hay camara';
    cam3='No  hay camara';
end

if ss==2
    cam1=IMAQHWINFO('winvideo',1)
    cam2=IMAQHWINFO('winvideo',2)
    cam3='No  hay camara';
end

if ss==3
    cam1=IMAQHWINFO('winvideo',1)
    cam2=IMAQHWINFO('winvideo',2)
    cam3=IMAQHWINFO('winvideo',3)
end

inf=get(hObject,'Value');

if inf==1
    set(handles.edit60,'string',['']);
    set(handles.cama1,'enable','off');
    set(handles.cama1,'visible','off');
    set(handles.cama2,'enable','off');
    set(handles.cama2,'visible','off');
    set(handles.cama3,'enable','off');
    set(handles.cama3,'visible','off');
elseif inf==2
    if ss==1
    set(handles.edit60,'string',[cam1.DeviceName]);
    end
    if ss<=0
     set(handles.edit60,'string',[cam1]);
    end
    set(handles.cama1,'enable','on');
    set(handles.cama1,'visible','on');
    set(handles.cama2,'enable','off');
    set(handles.cama2,'visible','off');
    set(handles.cama3,'enable','off');
    set(handles.cama3,'visible','off');
elseif inf==3
if ss==2
    set(handles.edit60,'string',[cam2.DeviceName]);
end
    if ss==1
    set(handles.edit60,'string',[cam2]);
    end

    if ss<=0
     set(handles.edit60,'string',[cam2]);
    end



    set(handles.cama1,'enable','off');
    set(handles.cama1,'visible','off');
    set(handles.cama2,'enable','on');
    set(handles.cama2,'visible','on');
    set(handles.cama3,'enable','off');
    set(handles.cama3,'visible','off');
else
    if ss==3
    set(handles.edit60,'string',[cam3.DeviceName]);
    end

    if ss==1
    set(handles.edit60,'string',[cam3]);
    end

    if ss==2
    set(handles.edit60,'string',[cam3]);
    end
    if ss<=0
     set(handles.edit60,'string',[cam3]);
    end


    set(handles.cama1,'enable','off');
    set(handles.cama1,'visible','off');
    set(handles.cama2,'enable','off');
    set(handles.cama2,'visible','off');
    set(handles.cama3,'enable','on');
    set(handles.cama3,'visible','on');
end
guidata(hObject,handles);


% --- Executes during object creation, after setting all
properties.
function camaras_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit60_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all
properties.
function edit60_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%_____________________________________________________
_____________________________
%_____________________________________________________
_____________________________
%activar camara
% --- Executes on button press in cama1.
function cama1_Callback(hObject, eventdata, handles)
set(handles.INICIO,'enable','on');
set(handles.INICIO,'visible','on');

handles.objVideo=videoinput('winvideo',1);%ACTIVAMMOS
EN EL OBJETO VIDEO LA CAMARA 1

guidata(hObject,handles);
axes(handles.axes3)
vidRes=get(handles.objVideo,'VideoResolution');
nBands=get(handles.objVideo,'NumberOfBands');
hImage=image(zeros(vidRes(2),vidRes(1),nBands));
preview(handles.objVideo,hImage);



% --- Executes on button press in cama2.
function cama2_Callback(hObject, eventdata, handles)
set(handles.INICIO,'enable','on');
set(handles.INICIO,'visible','on');

handles.objVideo=videoinput('winvideo',2);%ACTIVAMMOS
EN EL OBJETO VIDEO LA CAMARA 1
guidata(hObject,handles);
axes(handles.axes3)
vidRes=get(handles.objVideo,'VideoResolution');
nBands=get(handles.objVideo,'NumberOfBands');
hImage=image(zeros(vidRes(2),vidRes(1),nBands));
preview(handles.objVideo,hImage);

% --- Executes on button press in cama3.
function cama3_Callback(hObject, eventdata, handles)
set(handles.INICIO,'enable','on');
set(handles.INICIO,'visible','on');

handles.objVideo=videoinput('winvideo',3);%ACTIVAMMOS
EN EL OBJETO VIDEO LA CAMARA 1
guidata(hObject,handles);
axes(handles.axes3)
vidRes=get(handles.objVideo,'VideoResolution');
nBands=get(handles.objVideo,'NumberOfBands');
hImage=image(zeros(vidRes(2),vidRes(1),nBands));
```

```
preview(handles.objVideo,hImage);


% --- Executes on button press in stop.
function stop_Callback(hObject, eventdata, handles)
stop(handles.objVideo)%PARAMOS EL VIDEO
closepreview(handles.objVideo)%CERRAMOS EL PREVIEW


% --- Executes on button press in salir.
function salir_Callback(hObject, eventdata, handles)

fclose(handles.SerPIC);
%stop(handles.objVideo)%PARAMOS EL VIDEO
%closepreview(handles.objVideo)%CERRAMOS EL
PREVIEW
close %SALIRMOS DEL PROGRAMA

%_____________________________________________
_____________________
%_____________________________________________
_____________________


% --- Executes on button press in capturar.
function capturar_Callback(hObject, eventdata, handles)

imagen33=getsnapshot(handles.objVideo);
imwrite(imagen33, 'c:\imagen.jpg') ;



% --- Executes on button press in lectorp.
function lectorp_Callback(hObject, eventdata, handles)
warning off %#ok<WNOFF>



imagenl=imread('C:\leer.jpg');
% Show image
% Convert to gray scale
if size(imagenl,3)==3 %RGB image
   imagenl=rgb2gray(imagenl);
end
% Convert to BW
threshold = graythresh(imagenl);
imagenl =~im2bw(imagenl,threshold);
% Remove all object containing fewer than 30 pixels
imagenl = bwareaopen(imagenl,30);
%Storage matrix word from image


wordl=[ ];
rel=imagenl;
%Opens text.txt as file for write
fidl = fopen('text.txt', 'wt');
% Load templates
load templates
global templates
% Compute the number of letters in template file
num_letrasl=size(templates,2);
while 1
   %Fcn 'lines' separate lines in text
   [fll rel]=lines(rel);
   imgnl=fll;
   %Uncomment line below to see lines one by one
   %imshow(fl);pause(0.5)
   %------------------------------------------------------------------
   % Label and count connected components
   [Ll Nel] = bwlabel(imgnl);
   for nl=1:Nel
      [rl,cl] = find(Ll==nl);
      % Extract letter
      n1l=imgnl(min(rl):max(rl),min(cl):max(cl));
      % Resize letter (same size of template)
      img_rl=imresize(n1l,[42 24]);
      %Uncomment line below to see letters one by one
       %imshow(img_r);pause(0.5)
      %------------------------------------------------------------------
      % Call fcn to convert image to text
      letterl=read_letter(img_rl,num_letrasl);
      % Letter concatenation
      wordl=[wordl letterl];
   end
   %fprintf(fid,'%s\n',lower(word));%Write 'word' in text file
(lower)
   fprintf(fidl,'%s\n',wordl);%Write 'word' in text file (upper)
   % Clear 'word' variable

   wordl=[ ];
   %*When the sentences finish, breaks the loop
   if isempty(rel)  %See variable 're' in Fcn 'lines'
      break
   end
end
fclose(fidl);
%Open 'text.txt' file
winopen('text.txt')


% --- Executes on button press in capturap.
function capturap_Callback(hObject, eventdata, handles)
imagenleer=getsnapshot(handles.objVideo);
imwrite(imagenleer, 'c:\leer.jpg') ;



% --- Executes on button press in capturapiezas.
function capturapiezas_Callback(hObject, eventdata, handles)
imagenpiezas=getsnapshot(handles.objVideo);
imwrite(imagenpiezas, 'c:\i1.jpg') ;


% --- Executes on button press in sss.
function sss_Callback(hObject, eventdata, handles)
salir='s'
fprintf(handles.SerPIC,'%c','s');
```