TESIS DEFINITIVA. Incluye cambios solicitados por los revisores

SISTEMA AUTOMATIZADO DE RECONOCIMIENTO Y MANIPULACIÓN DE OBJETOS USANDO VISIÓN POR COMPUTADORA Y UN BRAZO INDUSTRIAL

Como requisito para obtener el grado de:

Maestro en Optomecatrónica

Presenta:

Ing. Alberto Martínez Rodríguez

Asesor:

Dr. J. Ascención Guerrero Viramontes

LEÓN, GUANAJUATO,

MÉXICO

Julio 2014

Johan,

Que mi legado sea luz en tu caminar…

Agradecimientos:

A mi asesor

Dr. J. Ascención Guerrero Viramontes

Por su paciencia, enseñanza, consejos y apoyo en el desarrollo del trabajo de investigación y en la elaboración de la tesis.

A mis revisores

Dr. J. Apolinar Muñoz Rodríguez

Dra. Ma. Del Socorro Hernández Montes

Por sus acertadas observaciones en el desarrollo del trabajo de tesis.

A CONACYT

Un agradecimiento al Consejo Nacional de Ciencia y Tecnología, por el soporte económico proporcionado durante los estudios de maestría.

Al Centro de Investigaciones en Óptica A.C.

Un agradecimiento especial a la institución que me dio la oportunidad de conocer excelentes amigos que laboraran en el centro y me abrió una puerta hacia el conocimiento y la ciencia

Resumen

Esta tesis tiene como propósito el exponer como pueden ser integrados a un robot industrial una serie de dispositivos electrónicos y algoritmos matemáticos para la clasificación, selección y almacenaje de objetos, piezas o productos de manufactura.
Los objetos que se usarán como ejemplos son unos bloques de madera con formas de: paralelepípedo, cubo, cilindro, y prisma triangular. Una webcam, toma una imagen de la mesa de trabajo, la imagen es analizada por la computadora y de esta manera se puede determinar la presencia previa de alguna pieza o lugares vacantes. Paso seguido, se dará la orden al robot de posicionar la pieza en el lugar que corresponda.
Un sistema de codificación integrado en los objetos se usa para guardar la información de forma, tamaño, peso, etc. De esta manera el sistema solo identifica la posición del objeto y la información codificada nos dice la forma y a donde se tiene que clasificar. Estos sistemas de codificación pueden ser a base de códigos bidimensionales QR o Dispositivos de lectura por radiofrecuencia.
Entonces, el sistema de visualización está activo para determinar la presencia de objetos y su posición. Al detectar una pieza, el sistema analiza la información y toma la decisión de cual posición le corresponde al objeto en el acomodo final.

Objetivos

- Lectura de información por medio de tecnología de identificación por radiofrecuencia (RFID por sus siglas en inglés: Radio Frequency Identification).
- Lectura de información por lectura de código de barras bidimensional llamado código QR (por sus siglas en inglés: *Quick Response Barcode)*.
- Diseño y construcción de la unidad de lectura automática y transmisión inalámbrica de datos hacia la computadora en protocolo Bluetooth.
- Identificación por medio de un procesamiento de imagen la presencia y posición de objetos en una plataforma.
- Generar un algoritmo para determinar el destino de una pieza en función de la lectura de sus datos y los resultados del procesamiento de imagen.
- Ejecución del acomodo de piezas en los lugares vacantes por un brazo robótico industrial.

Justificación

La gran demanda que existe en la actualidad en los sistemas de producción requiere de menos tiempo para la fabricación de un producto, por lo que la automatización de los procesos se vuelve algo fundamental en el desarrollo del sector industrial. La necesidad de automatizar ha dado como resultados la mejora de las herramientas y equipos que se utilizan; ya sea a nivel artesanal o industrial. En lo concerniente al ramo industrial, los sistemas de automatización, enfocados a los procesos de producción de alto riesgo ambiental, han obtenido en los robots industriales un gran aliado, los cuales han ido formando parte de las plantas productoras a gran escala.

Son muy comunes las tareas de manejo y suministro de material en las que el robot es una parte importante, pues se utiliza para transferir piezas a la entrada del proceso de producción o bien para recogerlas al final del mismo. Por tanto, es necesario que el robot pueda discriminar entre diferentes piezas: tamaños, formas o colores.

La discriminación de piezas, puede ser tan simple, que se sacrifique la confiabilidad del sistema; o tan exacta, que comprometa la velocidad de procesamiento. El presente proyecto propone un método de identificación independiente del objeto mismo. El método consiste en utilizar identificadores a modo de *etiquetas* en cada uno de dichos objetos. De ésta manera, la velocidad de discriminación de piezas se incrementa considerablemente sin comprometer la información exacta del objeto.
Asimismo, la información sobre el objeto se transmite de forma inalámbrica hacia la computadora.
La adaptación de sensores inalámbricos en la robótica industrial no solo tiene un impacto visual estético; la más grande aportación de un sistema escasamente cableado, es la seguridad. Una cantidad importante de tendidos de cables contribuye en gran medida, a tener una condición insegura de trabajo para el operador. También se podrán eliminar las fallas por conexiones defectuosas o envejecimiento de conductores, así como posibles capacitancias parásitas en los mismos, que arrojen falsas lecturas de los transductores. Finalmente, se evitan los procedimientos de instalación y los costos que implican.

Sus aplicaciones pueden ser muy diversas actualmente, por ejemplo en las embotelladoras, embutidos, compañías de reciclaje, etc., donde el proceso de selección puede ser muy repetitivo y en muchos de ellos se requiere de precisión y automatización. De esta manera, se puede evitar la intervención del hombre en ambientes hostiles, mejorando su calidad a nivel laboral, reduciendo el índice de accidentes y acelerando de forma segura la producción.
Cabe mencionar que las posibilidades de una identificación plena de un producto en cualquier momento en los formatos aquí manejados son muy grandes, ya que pueden incluir datos como:

- Fecha de elaboración.
- Fecha de caducidad.
- Número de serie.
- Lote.
- Línea de producción.
- Modelo.
- Cliente.
- Proceso de elaboración.
- Dimensiones.
- Peso.

- Material de fabricación.
- Etc.

En resumen, el presente proyecto resuelve la necesidad de contar con una identificación de piezas mucho más rápida y confiable, resuelve la complejidad computacional para la identificación de piezas, adicionalmente, la cantidad de información que se puede obtener sobre el objeto está muy por encima de los métodos convencionales de identificación. Resuelve la necesidad de la instalación de un cableado desde el punto geográfico de lectura hasta el computador, lo que contribuye a un mejor ambiente de trabajo y una flexibilidad total en cuanto al posicionamiento y movilidad de los componentes. También evita la realización de conexiones adicionales si una computadora adicional remota necesitase monitorear la información de los objetos, ya sea con fines de calidad, estadística o control de alguna otra área.
Finalmente, es necesario evitar paros en el proceso por culpa de colisiones en la plataforma destino. Por medio del procesamiento de imagen, el sistema es capaz de "ver" en dónde colocar la siguiente pieza.

Capítulo 1

Introducción

Las Innovaciones en la tecnología avanzan muy rápidamente, por lo que es posible el desarrollo de proyectos cada vez más completos y novedosos haciendo uso de nuevas tecnologías que se encuentran disponibles al público en general.

El desarrollo del presente proyecto implica el uso de código de barras bidimensional, identificación por radiofrecuencia, transmisión inalámbrica de datos, procesamiento por computadora y manipulación de objetos. Por lo que se hará uso de una webcam, un dispositivo de lectura/escritura para identificadores RFID, transmisor y receptor Bluetooth, una PC y un robot industrial.
El proyecto consiste en manipular varios objetos desde una posición fija inicial, para acomodarlos sobre una superficie plana. Dicha superficie es referida continuamente en ésta tesis como “Mesa de trabajo”. Para discriminar los objetos, se hace uso de símbolos QR y de dispositivos o Tag´s RFID adheridos a ellos. El primero de estos casos es por medio de captura y procesamiento de imagen, y el segundo utiliza tecnología y conceptos de radiofrecuencia.

Después de describir el proyecto, a lo largo de los capítulos se describen los antecedentes de cada parte del proyecto, y en su caso, la construcción y/o configuración de elementos.
Adicionalmente, se incluyen algunas aplicaciones prácticas que pudiera tener el presente proyecto; tanto industriales como civiles.
Finalmente, los apéndices proporcionan información detallada sobre aspectos específicos de los dispositivos, que caerían fuera del propósito de los capítulos que aquí se redactan. Sin embargo, pueden resultar útiles si se quiere tomar el presente trabajo como punto de partida para desarrollar futuras aplicaciones, o bien, perfeccionarlo.

1.1 Antecedentes

La robótica es un tema relativamente nuevo de estudio constante y progresivo en todo el mundo. Desde la consolidación de robots industriales como alternativas de operación económicas, eficientes, robustas y flexibles, y el estudio de sus arquitecturas; los aspectos de interés y estudio han sido hasta nuestros días proveer a estas máquinas con toda la información y herramientas necesarias para realizar su trabajo más rápido, preciso y seguro.

Un ejemplo de ello es el trabajo “A ROBOTIC SYSTEM FOR ROAD LANE PAINTING” de SangkyunWoo, Daehie Hong,Woo-Chang Lee, Jae-Hun Chung y Tae Hyung Kim en 2007.
Este trabajo trata el problema de repintar las líneas en las carreteras usando un robot móvil. Su importancia se centra en el uso de visión artificial para la detección de intervalos que requieran de pintura. Su utilidad contra la complejidad del proyecto es alta debido a que el robot móvil solo requiere de una cámara para realizar su trabajo. Implementa algunos filtros bastante conocidos para reconocer las líneas blancas del camino [Woo, 2006].

Otro ejemplo de una cámara como único sensor se encuentra en la tesis “DISEÑO DE ROBOT MÓVIL Y CONTROL MEDIANTE VISIÓN POR COMPUTADORA PARA LA NAVEGACIÓN EN UN

ESPACIO CONOCIDO." De José Manuel Sámano Molina, CIO 2013. Donde una computadora, determina los movimientos que debe seguir un robot móvil para evitar obstáculos en un espacio delimitado a partir de una imagen obtenida de dicho espacio.
Sin embargo, el tema no puede limitarse al uso de un solo sensor de imagen. Tal es el caso del trabajo titulado "SISTEMA DE VISIÓN PARA EL EQUIPO DE ROBOTS AUTÓNOMOS DEL ITAM", por Luis Alfredo Martínez Gómez, 2004. En el cual se hace uso de un sistema estereoscópico de visión para determinar la situación de un mini campo de futbol robótico. A partir de dicho procesamiento, se mandan las órdenes de acción para los autómatas participantes.

La visión artificial también se ha utilizado para encontrar determinar la su posición y orientación de un robot dentro de su entorno, en contraste con otras técnicas que utilizan, entre otros dispositivos, láseres, luz infrarroja, o técnicas de odometría para el mismo propósito.
Un ejemplo se encuentra en:" POSICIONAMIENTO DE ROBOTS BASADO EN VISION", de Marcano Gamero y Cosme Rafael, 2007. Donde se hace una revisión somera de algunos métodos de localización de robots basados en la visión computarizada. Estos métodos incluyen:
Uso de marcas de referencias fijadas a tierra, modelos de objetos, mapas y construcción de mapas basado en las características observadas.

Finalmente, se tiene la parte de reconocimiento de patrones, que "*es la categorización de datos de entrada en clases identificadas, por medio de la extracción de características significativas o atributos de los datos extraídos de un medio ambiente que contiene detalles irrelevantes*" [1].
Las metodologías que se utilizan para reconocimiento de patrones pueden agruparse en 4 grupos principales:
Heurísticas.- Se hace uso de la experiencia humana, por lo general están hechos a la medida del problema que se desea resolver.
Matemáticas.- Se basan en reglas de clasificación formuladas en un marco matemático, y se divide en dos categorías: Determinística y estadística. Entre los métodos determinísticos, podemos encontrar, por ejemplo, la clasificación por distancia euclídea. Las estadísticas consisten en representar cada patrón mediante un vector de números, y cada clase por uno o varios patrones prototipo.
Lingüísticas o sintácticas.- Se reduce un objeto a un conjunto de objetos estructurales o *primitivas*. Y se desarrolla una sintaxis para relacionar éstos elementos de forma espacial.
Redes neuronales artificiales.- El modo de análisis implica la configuración de una red de neuronas artificiales y el entrenamiento de la red para determinar cómo las neuronas individuales pueden afectar uno a la otra. El modo de reconocimiento implica el envío de datos a través de la red y la evaluación a que clase se aproximará más.

Como ejemplo de un reconocimiento de patrones por clasificación determinística, se puede ver la tesis "ALGORITMOS PARA SELECCIÓN Y CLASIFICACIÓN DE OBJETOS POR MEDIO DE BRAZO ROBOT Y VIDEOCÁMARA", por José Natividad Rodríguez Valtierra, CIO 2012.
Mientras que en el documento titulado "SISTEMA DE VISIÓN ARTIFICIAL PARA EL RECONOCIMIENTO Y MANIPULACIÓN DE OBJETOS UTILIZANDO UN BRAZO ROBOT", por Sobrado Partida Eddie, 2003, se tiene el ejemplo de un reconocimiento de patrones por medio de redes neuronales. Sin embargo, la manipulación de objetos se limita al llenado de tres contenedores que cuentan con una posición fija, y no se cuenta con un modo de retroalimentación que dé información sobre el resultado de los movimientos efectuados.

1.2 Descripción del proyecto

El esquema de la estructura del proyecto se muestra en la figura 1, donde se puede observar que la computadora recibe la información de las dos videocámaras y del módulo de lectura RFID, para luego, después de un proceso, enviar órdenes de movimiento al brazo robot.

El alcance del proyecto es: por medio del brazo robot, acomodar en una mesa de trabajo, cuatro distintos tipos de piezas. Se supone que éstas llegan por medio de una banda transportadora al punto de lectura marcado con rojo en la figura 1.
A los lados de dicho punto, estarán una cámara web (Webcam2) y un lector de RFID (Unidad RWD). Ambos tienen la misma función: Leer información de la pieza.
Por medio del brazo robótico, los objetos son tomados en el punto de lectura y se colocan en la mesa de trabajo. La superficie de dicha mesa está dividida en doce espacios, donde podrán colocarse las piezas.

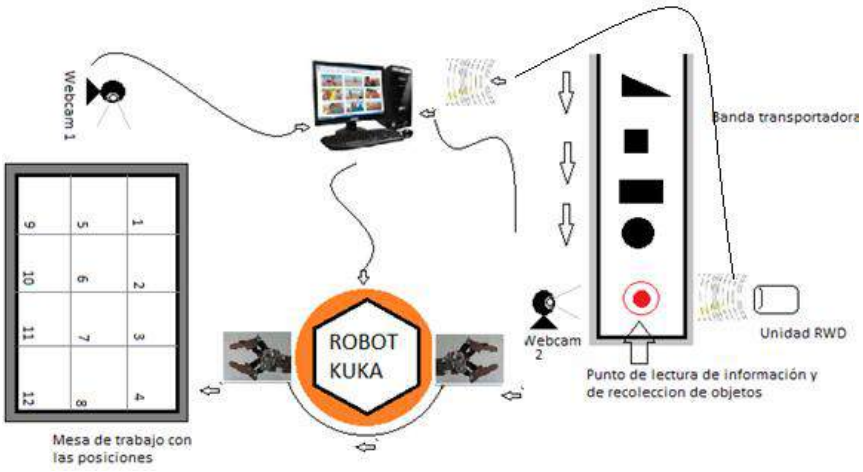

Figura 1.- Esquema general del proyecto

Para poder acomodar los objetos en la mesa de trabajo, es necesario que el sistema cuente con una forma de saber en qué lugares se pueden colocar. Es decir: identificar los lugares vacantes para evitar colisiones.
Ese objetivo se cumple colocando una cámara que tome la imagen de la superficie de la mesa (Webcam 1 en la figura 1). Acto seguido, por medio de un procesamiento de imagen en 2D, se determinará qué lugares ya se encuentran ocupados por algún otro objeto.
Las siguientes imágenes, ilustran el ambiente del desarrollo del proyecto (figuras 2-7). Las condiciones físicas reales son muy parecidas a la imagen de la figura 1, donde se observa al brazo robot KUKA en la parte central, entre la supuesta banda transportadora y la mesa de trabajo. En el modelo real, únicamente la computadora se encuentra alejada de los demás componentes; fuera del cerco de seguridad del laboratorio.

La Webcam 1 se encuentra montada en una estructura por encima de la mesa de trabajo, de manera que se tenga una imagen centrada de ésta. Se eligió de esa manera para tener un panorama constante de la imagen a procesar. De lo contario, se colocaría en el brazo robot, con un coste computacional muy elevado, al tener que hacer un procesamiento de imagen en 3D.

Además, aun tomando la imagen desde una posición fija del robot, se encuentra expuesta a desajustes o accidentes por el constante movimiento de éste.
La figura 2 muestra la Webcam 1, que es la encargada de tomar la imagen de la mesa de trabajo

Figura 2.- Webcam 1 montada en el muro para tomar la imagen de la mesa de trabajo

La figura 3 corresponde al robot industrial KUKA usado para la manipulación de objetos. En la misma imagen se aprecia la mesa de trabajo y algunos de los dichos objetos.

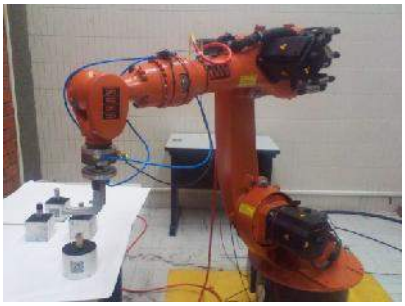

Figura 3.- Entorno de trabajo donde se aprecia al Robot KUKA junto a la mesa de trabajo con los Objetos

Los Objetos son bloques de madera con un asidero vertical colocado en su parte superior. Ese asidero sirve para que las pinzas del robot puedan sujetarlos y colocarlos en la mesa de trabajo. Los bloques pueden ver en la figura 4.

Los bloques también deberán tener fijados sus identificadores. El identificador QR consiste en una imagen del código de barras bidimensional protegida por una mica transparente. Mientras que el identificador RFID será un Tag del tipo tarjeta bancaria.

En la figura 5, se presenta el dibujo esquemático de un bloque en vista superior. Por un lado tendrán pegado el Tag RFID (en rojo en la figura 5), y por el otro lado, tendrán pegado su código QR (Gris en la figura 5). De ésta forma, el bloque puede ser identificado por cualquiera de las dos formas.

Figura 4.- Objetos con mango e identificadores QR y Tag´s tipo tarjeta, respectivamente

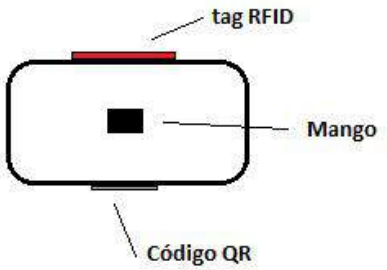

Figura 5.- Localización de los identificadores en el Objeto (vista superior).

En la figura 6 se muestra el punto de detección. En el lado izquierdo está la unidad de lectura RFID y por el lado derecho se encuentra una webcam (webcam2) encargada de leer los códigos QR. En la parte central está el espacio para que sea colocado el bloque como en la figura 7

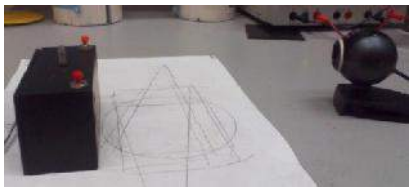

Figura 6.- Punto de identificación.

Figura 7.- Bloque en posición de identificación.

Interacción entre los componentes.

De acuerdo a la sección anterior, el sistema está compuesto por

- Unidad RFID
- Webcam 1 Acteck ATW-650
- Computadora de escritorio
- Interfaz USB
- Robot KUKA
- Webcam 2 Logitech modelo E300

En ésta sección se describirá la interacción entre todos éstos componentes, que se expresa también en la figura 8.

Iniciando de izquierda a derecha en dicha imagen, se tiene primeramente a los identificadores (Tag´s y códigos QR), luego, los lectores de los identificadores. Se trata de la Unidad RDW y la webcam 2 de la figura 1. El usuario, por medio de la interfaz programada en la PC, elegirá el modo preferido de operación. Es decir, si se usarán los códigos QR o los Tag´s como identificadores.

La información de los identificadores llegará a la computadora vía adaptador Bluetooth-USB o vía webcam 2.

Los datos recibidos por cualquiera de las dos vías, serán procesados por el software hecho en MATLAB. La computadora tendrá una imagen previa de la mesa de trabajo capturada por la webcam 1. Gracias al procesamiento de imagen, se determinará en dónde deberá colocar la pieza el robot. Esa orden de movimiento se manda vía USB a un módulo de salida, que se encargará de transformar la orden de la computadora, en niveles lógicos digitales para las entradas del robot. El esquema se ilustra en la figura 8.

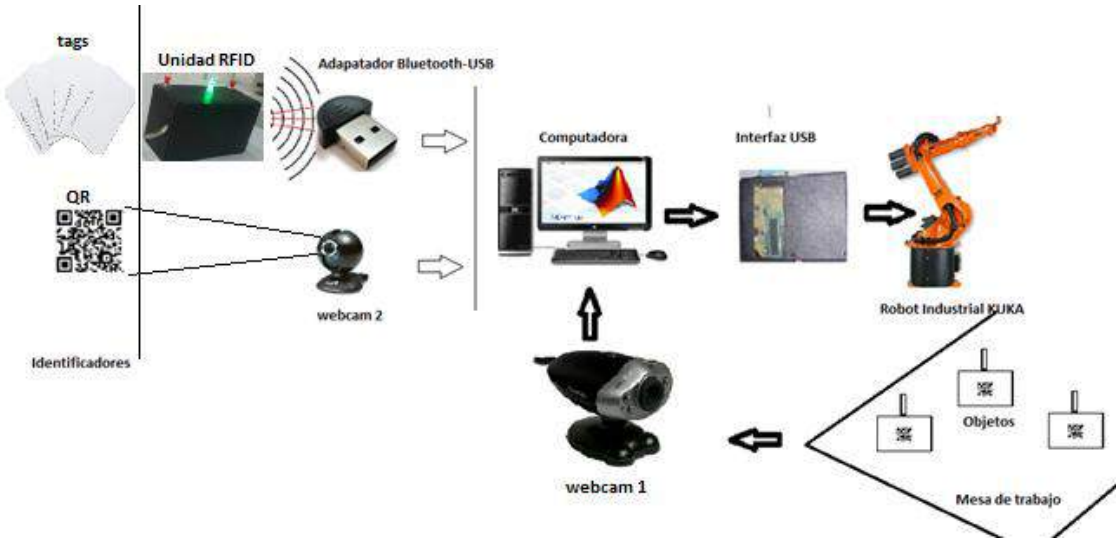

Figura 8.- Flujo de información entre los componentes

Una vez completado el movimiento del robot, la PC volverá a tomar la imagen de la mesa de trabajo para procesarla y actualizar su información en lo que a lugares disponibles corresponde.

Capítulo 2

Robótica

INTRODUCCION

Este capítulo menciona algunos aspectos históricos de ésta disciplina, resaltando algunos de sus principales logros a través del tiempo. Se mencionan los diferentes tipos de robot, sin embargo, tiene un enfoque marcadamente inclinado hacia los robots industriales; de los cuales se describen sus principales componentes, y se muestra un panorama sobre la utilización de robots industriales. Finalmente se abarcan los aspectos teóricos sobre la cinemática de un brazo robot.

2.1 Historia

El término ROBOT, proviene de la palabra checa *robota* y significa "trabajo". Fue introducido en nuestro vocabulario por el dramaturgo Karel Capek (1890-1938) en 1921 en su novela *Rossum´s Universal Robots,* donde describe al robot como una máquina que sustituye a los seres humanos para ejecutar tareas sin descanso; a pesar de esto, los robots se vuelven contra sus creadores aniquilando a toda la raza humana. Desde aquel entonces, prácticamente a cualquier sistema mecánico manipulador, se le llama robot [2].

En el término "Robot" confluyen las imágenes de máquinas para la realización de trabajos productivos y de imitación de movimientos y comportamientos de seres vivos [3].
Los robots actuales son obras de la ingeniería y como tales concebidas para producir bienes y servicios o explotación de recursos naturales.
Sin embargo, desde la antigüedad el hombre ha sentido fascinación por las máquinas que imitan la figura y el movimiento de seres animados. Existe una larga tradición de autómatas desde el mundo griego hasta nuestro siglo, pasando por los autómatas de los artesanos franceses y suizos del siglo XVII que ya incorporaban interesantes dispositivos mecánicos para el control automático de movimientos [3]. La tabla 1, muestra un poco la historia de la robótica [4].
En nuestro siglo, el desarrollo de máquinas ha estado fuertemente influido por el progreso tecnológico.

AÑO	MAQUINA
siglo XVII	J. Vaucanson construyo varias muñecas mecánicas que ejecutaban pieza de música.
1801	J. Jacquard inventó su telar, que era una máquina programable para la urdimbre
1805	H. Maillardet construyó una muñeca mecánica capaz de hacer dibujos
1946	El Norteamericano G.C. Devol desarrollo un dispositivo controlador que podía registrar señales eléctricas por medios magnéticos y reproducirlas para accionar una maquina mecánica.
1951	Trabajo de desarrollo con tele operadores para manejar materiales radioactivos. Patente concedida a Goertz y Bergsland E.E.U.U.

1959	Introducción del primer robot comercial por Planet Corp. Estaba controlado por interruptores de fin de carrera y levas.
1966	Trallfa, una firma noruega, construyó e instaló un robot de pintura por pulverización.
1968	Un robot móvil se desarrolló en SRI (Instituto de investigación de Stanford). Estaba provisto de una diversidad de sensores, incluyendo una cámara de visión y sensores táctiles y podía desplazarse sobre el suelo.
1973	Se desarrolló en ese mismo instituto el primer lenguaje de programación por computadora para control de robot.
1975	El robot "SIGMA", de Olivetti se utilizó en operaciones de montaje, una de las primitivas aplicaciones de la robótica al montaje.
1979	Desarrollo del robot tipo SCARA en la universidad de Yamanashi en Japón para montaje. Varios robots SCARA comerciales se introdujeron hacia 1981.
1984	Varios sistemas de programación fuera de línea se demostraron en la exposición Robots 8. La operación típica de estos sistemas permitía que se desarrollaran programas de robot utilizando gráficos interactivos en una computadora personal y luego se cargaban al robot.

Tabla 1.- Principales logros del desarrollo de la robótica

Adicionalmente, en 1954 Devol (ver tabla 1) concibió la idea de un dispositivo de transferencia programada de artículos [4] y dos años más tarde comparte la idea con Joseph F. Engelberger, director de ingeniería de la división aeroespacial de la empresa Manning Maxwell y Moore en Stanford y juntos comienzan a trabajar en la utilización industrial de sus máquinas fundando la Consolidated Controls Corporation, que más tarde se convertiría en Unimation (Universal Automation), e instalando su primera máquina Unimate en la fábrica de General Motors de Trenton, New Jersey, en una aplicación de fundición por inyección, en 1960.

2.2 Clasificación

La robótica se ha convertido en un área clave y estratégica para todo país en desarrollo, es sinónimo de modernización y coadyuva a proporcionar bienestar a la sociedad. La figura 9 muestra un ejemplo de un robot industrial de la compañía alemana KUKA.

Los robots pueden realizar aplicaciones de alto impacto en la sociedad, por ejemplo, fisioterapia asistida por robótica donde el paciente recobra la movilidad de sus extremidades con mayor facilidad, eficiencia y menor tiempo. Particularmente, para un cierto sector de la sociedad, específicamente personas con capacidades diferenciadas, los robots pueden incrementar la calidad de vida.

Figura 9.- Robot Industrial marca KUKA

El uso de la robótica en la medicina juega un papel destacado, ya que se convierte en una eficiente herramienta que permite incrementar la seguridad y exactitud en la ejecución de cirugías de alto riesgo [5], así como su realización a través de la distancia. La primera intervención de éste tipo fue llevada a cabo en España en 1998 [6].

Actualmente existe una gran variedad de robots con diversas estructuras geométricas y mecánicas que definen su funcionalidad y aplicación. Sin embargo, de manera general pueden ser clasificados como se muestra en la tabla2 [2].

Clasificación de robots		
Móviles	Terrestres: ruedas, patas	
	Submarinos, Aero-espaciales	
Humanoides	Diseño Complejo	
Industriales	Brazos Mecánicos	Robots Manipuladores

Tabla 2.- Clasificación de los robots

Los robots industriales son el tipo más popular de robot, debido a la importancia que ocupan en el sector industrial como herramientas clave para la modernización de las empresas. Hoy en día, la automatización de procesos industriales es realizada a través de robots y esto trae como consecuencia competitividad, productividad, eficiencia y rentabilidad de las empresas.

La Organización internacional para la estandarización (ISO), define el robot industrial como “un manipulador multipropósito, reprogramable y controlado automáticamente en tres o más ejes”. [2].

Los robots Industriales también son conocidos como brazos robot o brazos mecánicos, por analogía con el brazo humano. Un ejemplar de esta naturaleza puede llegar a pesar alrededor de 3 toneladas y puede alcanzar una altura de 4 metros y velocidad de movimiento de 3000mm/s. [2].

Dentro de las características de los robots industriales se encuentran el que trabaja sin descansar 24 horas al día, todos los días del año, por lo que en aplicaciones industriales, superan en

desempeño a las personas, ya que pueden repetir el proceso siempre en el mismo tiempo y con la misma calidad.

Entre las compañías más importantes que diseñan y construyen los robots industriales se encuentran FANUC, ABB, KUKA, MOTOMAN, EPSON; y cuentan con una gran diversidad de modelos de robots para diferentes aplicaciones industriales.

Las principales aplicaciones que actualmente tienen los robots industriales están involucradas con el proceso de pintado de carrocerías automotrices, accesorios, cubetas, tinas, cajas, soldadura de punto y por arco en carrocerías automotrices, puertas y diversas piezas industriales; traslado de herramientas, estribado y empaquetado de materiales, etc. [2].

En forma general, un robot industrial está formado por los siguientes elementos:

- Articulaciones o uniones formadas por servomotores que permiten la conexión y movimiento relativo entre dos eslabones consecutivos de un robot. Pueden ser del tipo *rotacional* o *lineal*. Las articulaciones tipo lineal (figura 10), también son conocidas como prismáticas. Las unidades de medición relacionadas con una articulación rotacional (figura 11) están dadas en radianes o grados, mientras que para una lineal, generalmente se encuentran en metros.

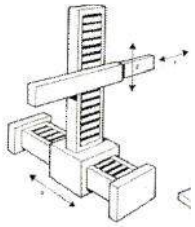

Figura 10.- Articulación Lineal

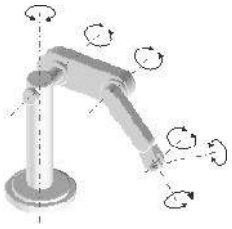

Figura 11.- Articulaciones rotacionales

- Actuadores: son dispositivos capaces de transformar energía hidráulica, neumática o eléctrica en la activación de un proceso con la finalidad de generar un efecto. Pueden ser servomotores, elementos neumáticos, eléctricos o hidráulicos.
- Sensores: proporcionan información del estado interno del robot. Posición y velocidad articular, que son las variables más comunes en los sistemas de sensores
- Sistema mecánico: consiste en la secuencia de eslabones rígidos conectados en cadena abierta por medio de articulaciones.

- Consola de control: se compone de un sistema electrónico con la etapa de potencia encargada de suministrar energía al robot para su movimiento. Incluye un dispositivo portátil llamado *teach pendant*, el cual brinda la interfaz necesaria para que el usuario se comunique con el robot a través de instrucciones de programación

Cada vez es mayor el número de robots industriales que se usan en las empresas de todo el mundo, en particular, las líneas de producción son operadas por medio de robots manipuladores.

Desde el año 2000 se ha dado una tasa anual de crecimiento sistemático de 25 000 robots, y los principales usuarios son: Estados Unidos, Japón, Alemania, Italia, Francia y China. Para el año 2008 se encontraban operando en todo el mundo más de un millón de robots industriales [2].

En la figura 12, se muestra el número de robots industriales por cada 10mil trabajadores en diferentes regiones del mundo [7].

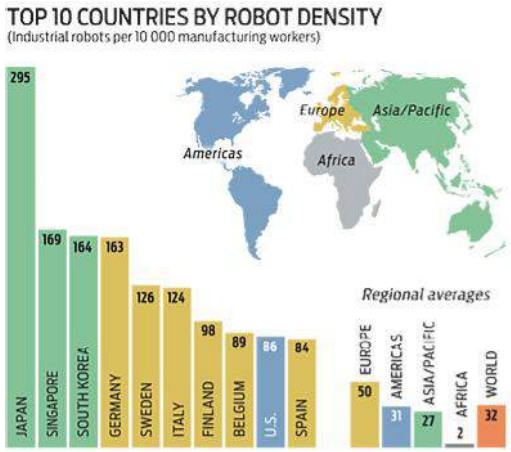

Figura 12.- Cantidad de robots industriales por cada 10mil trabajadores de manufactura.

Se observa que Japón reporta una gran diferencia respecto a los demás países; si bien es cierto que se trata de un país vanguardista, también es verdad que ellos cuentan como robots a máquinas que, en otras partes del mundo se considerarían simplemente como "máquinas de fábrica" [8].

2.3 Sistema básico de un robot manipulador

Un robot manipulador operando individualmente necesita como mínimo los siguientes componentes:

a) El Brazo (robot) consiste en un sistema de articulaciones mecánicas (eslabones, engranajes, transmisión por cadena o por correa), actuadores (motores eléctricos o hidráulicos) y sensores de posición usados en el sistema de control de lazo cerrado.

b) El controlador, generalmente basado en microcomputador, que recibe las señales de los sensores de posición y envía comandos a la fuente de potencia controlada.
c) La unidad de conversión de potencia que alimenta a los motores que mueven las articulaciones.

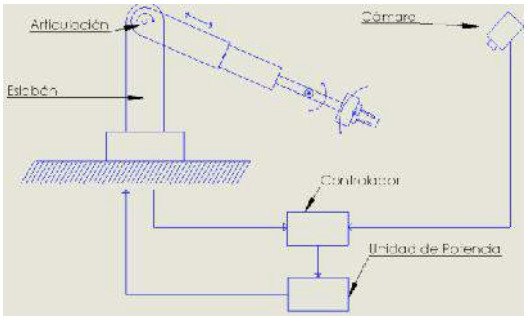

Figura 13 .- Componentes básicos de un sistema robot manipulador

Dependiendo del tipo de aplicación, un robot puede contar también con sensores externos, de los cuáles, el más poderoso es un sistema de visión consistente en una cámara de video, interfaz y computador procesador de imágenes. La información obtenida por éste sistema pasa al microcomputador que controla el robot y es usada para dirigir los movimientos de éste.

2.4 Cinemática de robots manipuladores

La cinemática estudia el movimiento (en éste caso, de estructuras mecánicas multi-articuladas), pero no la manera de controlar éste movimiento, que corresponde a la dinámica. Para el control de un brazo manipulador se necesita, además de su descripción cinemática, una formulación matemática de su dinámica. Una excepción ocurre cuando el control del brazo se efectúa por medio de redes neuronales.

Un brazo manipulador tiene varios grados de libertad. El movimiento de cada eslabón puede referirse al sistema de coordenadas del eslabón anterior, partiendo desde la "mano" hacia la "base".

En ésta forma es posible describir el movimiento deseado de la herramienta de trabajo en función del movimiento de cada una de las articulaciones. Para ese objetivo, se necesitan definir los *parámetros de eslabón* y en base a ellos, las *matrices homogéneas* de cada eslabón y del espacio de trabajo.

Cada eslabón tiene su propio sistema de coordenadas, con el origen en la articulación que controla el movimiento. El movimiento puede ser rotatorio o traslatorio. El eslabón "i" va desde la articulación "i" hasta la "i+1", con el origen del sistema de coordenadas "i" sobre la articulación "i+1". El movimiento de los eslabones debe de estar relacionado con un sistema de referencia inercial, que se elige generalmente en la base del robot. En esta forma se puede obtener una relación entre una tarea específica caracterizada por una *matriz de transformación de tarea* y las matrices de eslabón correspondientes. Éstas últimas determinan el movimiento necesario de cada eslabón para efectuar la tarea en cuestión.

Se mostrarán las transformaciones generales entre sistemas de coordenadas. También cómo las matrices homogéneas de transformación permiten describir por medio de una sola matriz 4X4 traslación y rotación, es decir, posición y orientación. Finalmente, por medio de los parámetros de eslabón se determinarán las matrices homogéneas de eslabón

Una simple multiplicación matricial permite, entonces, relacionar las matrices de eslabón a la matriz homogénea de una tarea determinada.

Rotación de un sistema de coordenadas

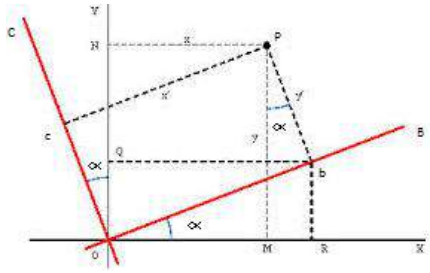

Figura 14.- Un sistema ABC girado con respecto a otro XYZ en α grados

En la figura 14 el sistema ABC se hace rotar respecto al sistema XYZ, que se considera como sistema de referencia. Primero la rotación se hace sobre el eje X en un ángulo α. Un punto P con componentes "a, b, c" en el sistema ABC se representa por el vector

$$\boldsymbol{P} = a\boldsymbol{i} + b\boldsymbol{j} + c\boldsymbol{k} \qquad \textbf{ec. 1}$$

Donde i, j, k son los vectores unitarios sobre los ejes A, B, C respectivamente. Para obtener el punto **P** en coordenadas X, Y, Z se utiliza el producto escalar de **P** con los vectores unitarios en el sistema X, Y, Z i_x, j_y, k_z.

$$\boldsymbol{x} = \boldsymbol{P} \cdot \boldsymbol{i_x} = \boldsymbol{a} \qquad \textbf{ec. 2}$$

$$\boldsymbol{y} = \boldsymbol{P} \cdot \boldsymbol{j_y} = \boldsymbol{b}\cos\alpha - \boldsymbol{c}\,sen\alpha \qquad \textbf{ec. 3}$$

$$\boldsymbol{z} = \boldsymbol{P} \cdot \boldsymbol{k_z} = \boldsymbol{b}\,sen\alpha - \boldsymbol{c}\cos\alpha \qquad \textbf{ec. 4}$$

Este resultado puede escribirse en forma matricial, como se indica en la ecuación 5. En ella, la matriz de la derecha recibe el nombre de *matriz de rotación.*

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -sen\alpha \\ 0 & sen\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \qquad \textbf{ec. 5}$$

Puede obtenerse una expresión más general para una rotación arbitraria del sistema ABC con respecto al sistema XYZ, siempre que los cosenos direccionales entre los dos sistemas sean conocidos. El resultado es:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} i_a \bullet i_x & i_b \bullet i_x & i_c \bullet i_x \\ i_a \bullet i_y & i_b \bullet i_y & i_c \bullet i_y \\ i_a \bullet i_z & i_b \bullet i_z & i_c \bullet i_z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \qquad \textbf{ec. 6}$$

En la expresión anterior, la matriz 3x3 de productos escalares es la *matriz general de rotación*.

Cada columna de ésta matriz representa los componentes de los vectores unitarios $\mathbf{i_a}$, $\mathbf{j_b}$, $\mathbf{k_c}$ en la dirección de los ejes X, Y y Z del sistema XYZ. Cada hilera o columna en la matriz general de rotación representa un vector unitario normal a los vectores representados por las otras dos hileras o columnas, respectivamente. Esto significa que la inversa de una matriz de rotación es igual a su traspuesta,

$$\boldsymbol{R^{-1}} = \boldsymbol{R^t} \qquad \textbf{ec. 7}$$

$$R_{Y,\beta} = \begin{bmatrix} cos\beta & 0 & sen\beta \\ 0 & 1 & 0 \\ -sen\beta & 0 & cos\beta \end{bmatrix} \qquad \textbf{c. 8}$$

$$R_{Z,\gamma} = \begin{bmatrix} cos\gamma & -sen\gamma & 0 \\ sen\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \textbf{ec. 9}$$

Cuando se efectúan varias rotaciones con respecto a los ejes del sistema de referencia XYZ, la matriz total se obtiene *premultiplicando* la primera matriz de rotación por la segunda y así, sucesivamente,

$$R_{tot} = R_n + R_{n+1} \ldots \ldots R_2 R_1 \qquad \textbf{ec. 10}$$

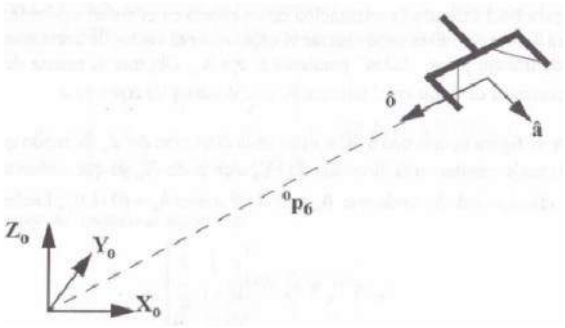

Figura 15.- Definición de los vectores $\hat{a}$ y $\hat{o}$ de la mano

Ahora, se desea obtener una interpretación de rotación en función de la orientación de la herramienta, que en muchos casos es una tenaza para coger y manipular objetos. Con referencia a la figura 15, se define el vector *de acercamiento* $\hat{a}$, como un vector unitario en la dirección del movimiento de la mano. El vector de *orientación* $\hat{o}$, se define como un vector unitario perpendicular tanto a $\hat{a}$, como a los "dedos". El sistema de coordenadas de la mano se completa con el vector unitario $\hat{u}$, que forma un sistema de mano derecha con $\hat{a}$ y $\hat{o}$, de modo que:

$$\hat{u} = \hat{o} \; x \, \hat{a} \qquad \textbf{ec. 11}$$

El origen de este sistema se coloca en el centro geométrico de la mano, usualmente. Si se desea referir las componentes de $\hat{a}, \hat{o}, \hat{u}$ al sistema de coordenadas de referencia (el de la base), podemos hacerlo por medio de una matriz de rotación dada por:

$$\boldsymbol{R} = \begin{bmatrix} u_x & o_x & a_x \\ u_y & o_y & a_y \\ u_z & o_z & a_z \end{bmatrix} \qquad \textbf{ec. 12}$$

Donde los elementos de **R** son componentes de $\hat{u}, \hat{o}, \hat{a}$ en el sistema de coordenadas de base.

Traslación de un sistema de coordenadas

La traslación del sistema de coordenadas con respecto a otro se representa por medio de la suma de vectores. Con referencia a la figura 16, si los sistemas ABC y XYZ coinciden inicialmente y se traslada ABC con respecto a XYZ, manteniendo los ejes A,B,C paralelos a XYZ respectivamente en el punto

$$\boldsymbol{P}_{abc} = \boldsymbol{a}i_A + \boldsymbol{b}j_B + \boldsymbol{c}k_C \qquad \textbf{ec. 13}$$

se convierte en

$$\boldsymbol{P}_{xyz} = (a + d_1)i_x + (b + d_2)j_y + (c + d_3)k_z \qquad \textbf{ec. 14}$$

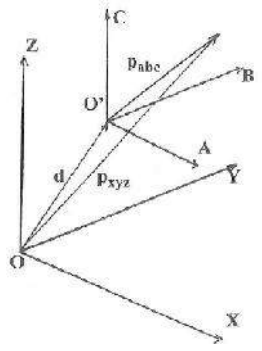

Figura 16.- Traslación del sistema ABC,con respecto a XYZ

Expresando esta relación en forma matricial con $\boldsymbol{P_X}$, $\boldsymbol{P_y}$ $y\boldsymbol{P_z}$ expresados como componentes de un vector columna, tendremos:

$$P_{XYZ} = \begin{bmatrix} P_X \\ P_Y \\ P_Z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & d_1 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} \qquad \textbf{ec. 15}$$

La matriz 4X4 es una matriz de traslación, $[T_r]$ y se puede escribir

$$\boldsymbol{P_{XYZ}} = [T_r]\boldsymbol{P_{ABC}} \qquad \textbf{16}$$

Donde

$$T_r = \begin{bmatrix} 1 & 0 & 0 & d_1 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \textbf{ec. 17}$$

Y es una *matriz homogénea de traslación.*

La tenaza o mano llega a la posición y orientación indicadas por medio de una rotación seguida de una traslación. Estas operaciones se efectúan con respecto al sistema de referencia, de modo que, para obtener la matriz total que incluye ambas operaciones, se debe premultiplicar la matriz que representa la primera por la matriz que representa la segunda.,

$$T_m = T_{rm}\, R_{rm} = \begin{bmatrix} u_x & o_x & a_x & p_x \\ u_y & o_y & a_y & p_y \\ u_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} [R] & P \\ 0 & 1 \end{bmatrix} \qquad \textbf{ec. 18}$$

Parámetros de eslabón

En un robot manipulador, una articulación "i" conecta dos eslabones, el eslabón "i-1" y el eslabón "i", como se ve en la figura 15. El eslabón cero es la base estática del robot y se usa como referencia, pero no aparece explícitamente en los parámetros de eslabón.

Los eslabones móviles se numeran del 1 al "n". Con referencia a la figura 15, el eslabón "i" es controlado por la articulación "i" y su movimiento es referido a la articulación anterior, el "i-1".

Un robot con "n" grados de libertad tiene **n+1** eslabones, contando el número cero, y **n+1** sistemas de coordenadas, también contando el de referencia y el de herramienta.

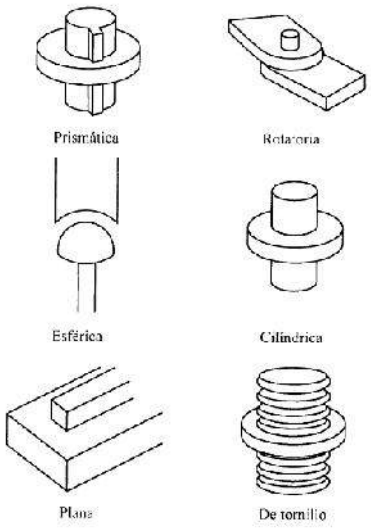

Figura 17.- Los seis tipos de articulaciones más comunes

Para determinar una matriz de transformación de eslabón se debe primero determinar un número necesario y suficiente de parámetros que describan la relación entre dos eslabones. En general, una matriz de transformación de eslabón se designa como ${}_{i}^{i-1}A$, indicando que es la transformación de puntos dados en el sistema "i" al sistema "i-1".

En la figura 17 se muestran los tipos de eslabones.

Se puede decir entonces que, para obtener una matriz de transformación ${}_{i}^{i-1}A$, podemos

a) Determinar qué operaciones son necesarias para mover el sistema {i} a la posición / orientación deseada partiendo de coincidencia con el sistema {i-1}
b) Determinar las operaciones necesarias para llevar el sistema {i-1} a coincidencia con el sistema {i} que se encuentra en la posición / orientación deseadas.

En ambos casos, si un punto P_i está dado en el sistema el {i} podremos obtener las coordenadas de este punto en el sistema {i-1} mediante

$$\boldsymbol{P_{i-1} = {}_{i}^{i-1}A\, P_i} \qquad \textbf{ec. 19}$$

[9]

Capítulo 3

Códigos de Barras

INTRODUCCION

El presente capítulo habla sobre la historia y evolución de los códigos de barras. Haciendo énfasis principalmente en la descripción del código QR, sus presentaciones, capacidades, así como procesos de codificación y decodificación.
Entre las características de la codificación QR, destacan: 40 versiones (tamaños) de un código QR, y la flexibilidad de poder ser leídos aún si la imagen está algo distorsionada, o incluso incompleta.

3.1 Historia y Clasificación

En 1948, Bernard Silver, estudiante graduado del Instituto Tecnológico de Drexel en Filadelfia, escuchó a un ejecutivo de supermercados solicitar algún método para capturar la información de algún producto automáticamente al momento de pagar.
Silver llegó a mencionar el problema a Norman Joseph Woodland, quien era profesor de ingeniería mecánica en la misma institución
Luego de trabajar en algunas ideas preliminares, estaban convencidos de que podrían crear un producto viable.
El 20 de octubre de 1949, Woodland y Silver presentaron una solicitud de patente titulada "Aparato y método de clasificación." Los inventores describieron su invención como relativa "a la técnica de clasificación a través del medio de identificación de patrones" [10].
La simbología se compone de un patrón de cuatro líneas blancas sobre un fondo oscuro. La figura 18 es la imagen de una página de dicha solicitud de patente. La primera línea era una línea de referencia y las posiciones de los restantes tres líneas fijas con respecto a la primera línea. La información fue codificada por la presencia o ausencia de una o más de las líneas. Esto permitió 7 diferentes clasificaciones de los artículos. Sin embargo, los inventores observaron que si hay más líneas añadidas, más clasificaciones podrían ser codificadas. Con 10 líneas, se podían codificar 1023 clasificaciones.
La solicitud de patente Woodland y Silver se publicó el 07 de octubre 1952 como Patente de EE.UU. 2.612.994

Durante los años 60's inició la investigación y el desarrollo de procesos de automatización de los puntos de venta en los supermercados, de tal manera que para principios de los 70's, se comenzó a operar con códigos de barras. Esto impulsado por la iniciativa de las grandes tiendas de autoservicio y gigantes de la computación. Después del desarrollo de varias simbologías, las computadoras permitieron que el código de barras fuera aceptado ampliamente. Hoy en día es indispensable para levantar inventarios, agilizar ventas, llevar control de inventarios, control de ventas, control de almacén, localización de productos, etc. [10].

Los códigos de barras se dividen en dos grandes grupos: los códigos de barras lineales y los códigos de barras de dos dimensiones [11].

Los códigos de barras lineales son Códigos basados en la representación mediante un conjunto de líneas paralelas verticales de distinto grosor y espaciado que en su conjunto contienen una determinada información, es decir, las barras y espacios del código representan pequeñas cadenas de caracteres. La figura 19 ejemplifica un código de barras lineal.

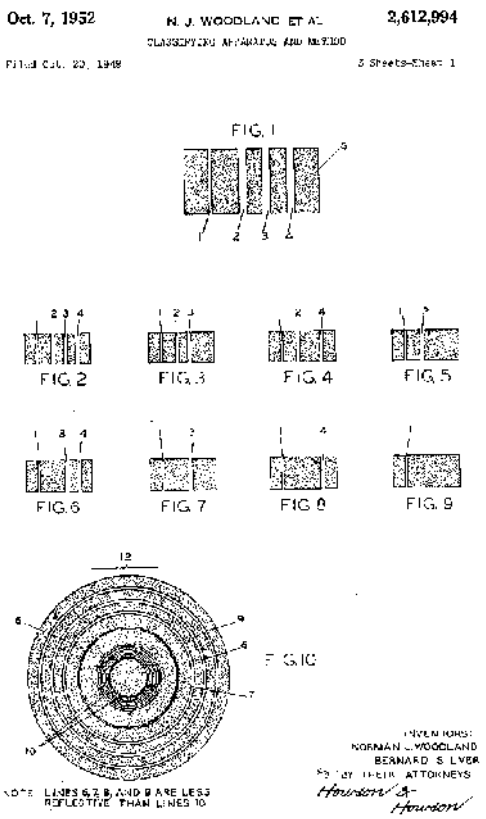

Figura 18.- Fragmento de la solicitud de patente presentada por Woodland y Silver en 1949, donde se muestra la simbología del código.

Como ejemplos de codificaciones lineales tenemos los estándares: EAN, Code128, Code39, Code93 y el Codabar.

Figura 19.- Código de barras lineal

Los códigos de barras bidimensionales son códigos basados en la representación mediante una matriz de datos (Figura 20). Por ejemplo, PDF417, Datamatrix, o QR. Los códigos bidimensionales contienen una mayor capacidad de datos comparado con los lineales (100 veces o más), lo que, en un principio, puede representar un tiempo mayor de procesamiento de datos.

Figura 20.-Códigos de barras Bidimensionales

Códigos basados en óptica

Los códigos de barras pueden codificar una gran cantidad de información numérica o en forma de texto. Sin embargo, existen otras formas de codificación que son capaces de contener imágenes, tales como los códigos basados en óptica. Dichos códigos son utilizados en la transmisión digital de datos, donde recientemente, la seguridad de los datos está cobrando una creciente importancia.

Esos métodos incluyen: marca de agua, técnicas basadas en caos, codificación fractal y de fase, holografía digital, transformada fraccional de Fourier, OR exclusiva óptica y polarización.

El método de marca de agua segmenta la imagen original en bloques. Luego, los bloques son transformados en secuencias en zigzag en el dominio de la frecuencia para obtener la imagen codificada.

Los métodos basados en fractales generan la imagen secreta mapeando la imagen original en bloques basados en una transformación afín. Este método recupera la imagen original detectando la fase vía transformada inversa. Estos métodos generan la imagen encriptada por medio de un patrón aleatorio, el cual es obtenido por una transformación en el dominio de la frecuencia. [11]

Los métodos restantes utilizan arreglos ópticos específicos como lentes, fuentes de luz, espejos, filtros espaciales, rejillas, etc. Otros métodos completamente computacionales consisten en que la encriptación se realiza por medio de un patrón de franjas generado computacionalmente. Luego, se sobrepone una máscara aleatoria al patrón de franjas, que se deforma de acuerdo a la intensidad de la imagen [12]. Alternativamente, existe otro método basado en una función triangular amortiguada y una secuencia discreta que se utilizan para generar un patrón aleatorio a partir de la imagen original [11].

3.2 Codigo QR

Un **código QR** (*Quick Response Barcode*) es un sistema para almacenar información en una matriz de puntos o un código de barras bidimensional creado por la compañía japonesa Denso-Wave en 1994 [13]; Quien ha decidido no ejercer sus derechos de autor. Fue desarrollado como una fusión de la alta densidad de información del formato PDF417 y la rápida lectura del código maxi. La integración del código QR se muestra en la figura 21.

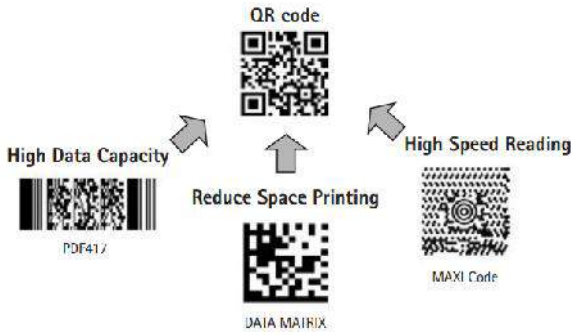

Figura 21.-.- Desarrollo del código QR

La codificacion QR es una simbología matricial que consiste en un arreglo de modulos cuadrados nomilales en un area cuadrada. Incluye un identificador de patron en tres de las cuatro esquinas del código, y su función consiste en proporcinar una facil localización de la posición, tamaño e inclinación (Figura 22)según [13]. Incluye una amplia gama de tamaños de simbolo y cuatro niveles de corrección de errores [14]. Sus aplicaciones se han vuelto populares como se ejemplifica en la figura 23.

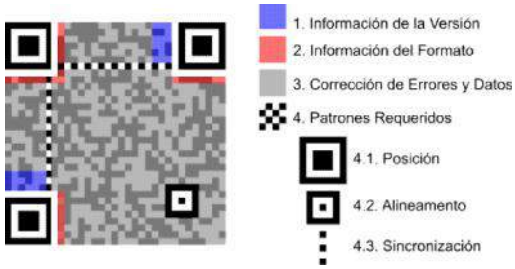

Figura 22.-Parámetros del símbolo QR.

Figura 23.-Una aplicación del código QR

La codificacion QR es una simbologia matricial con 4 niveles de corrección de errores, varios tamaños estandarizados, que van desde 21x 21 celdas, hasta 177 x 177 celdas, y 4 tipos de información. Éstas y más especificaciones se muestran en la tabla 3:

Symbol size	Min. 21x21 cell - Max. 177x177 cell (with 4-cells interval)	
Information type and volume	Numerical characters	7,089 characters at maximum
	Alphabets, signs	4,296 characters at maximum
	Binary (8 bit)	2,953 characters at maximum
	Kanji characters	1,817 characters at maximum
Conversion efficiency	Numerical characters mode	3.3 cells/character
	Alphanumerical/signs mode	5.5 cells/character
	Binary (8 bit) mode	8 cells/character
	Kanji character mode (13 bit)	13 cells/character
Error correction functionality	Level L	Approx. 7% of the symbol area restored at maximum
	Level M	Approx. 15% of the symbol area restored at maximum
	Level Q	Approx. 25% of the symbol area restored at maximum
	Level H	Approx. 30% of the symbol area restored at maximum
Linking functionality	Possible to be divided into 16 symbols at maximum	

Tabla 3.- Caracteerísticas de la simbología matricial según el Comité de Estándares de Tecnologías de la Información (ITSC), organismo regulador, con sede en Singapur.

La versión del símbolo (dimensión) se especifica por el usuario, de acuerdo a los estándares establecidos en la tabla 4. Las figuras 24 y 25 muestran un ejemplo de matriz para los tamaños mínimo y máximo respectivamente.

Version	No. of Modules/ side (A)	Function pattern modules (B)	Format and Version Information modules (C)	Data modules except (C) ($D=A^2-B-C$)	Data capacity [codewords][a] (E)	Remainder Bits
1	21	202	31	208	26	0
2	25	235	31	359	44	7
3	29	243	31	567	70	7
4	33	251	31	807	100	7
5	37	259	31	1 079	134	7
6	41	267	31	1 383	172	7
7	45	390	67	1 568	196	0
8	49	398	67	1 936	242	0
9	53	406	67	2 336	292	0
10	57	414	67	2 768	346	0
11	61	422	67	3 232	404	0
12	65	430	67	3 728	466	0
13	69	438	67	4 256	532	0
14	73	611	67	4 651	581	3
15	77	619	67	5 243	655	3
16	81	627	67	5 867	733	3
17	85	635	67	6 523	815	3
18	89	643	67	7 211	901	3
19	93	651	67	7 931	991	3
20	97	659	67	8 683	1 085	3
21	101	882	67	9 252	1 156	4
22	105	890	67	10 068	1 258	4
23	109	898	67	10 916	1 364	4
24	113	906	67	11 796	1 474	4
25	117	914	67	12 708	1 588	4
26	121	922	67	13 652	1 706	4
27	125	930	67	14 628	1 828	4
28	129	1 203	67	15 371	1 921	3
29	133	1 211	67	16 411	2 051	3
30	137	1 219	67	17 483	2 185	3
31	141	1 227	67	18 587	2 323	3
32	145	1 235	67	19 723	2 465	3
33	149	1 243	67	20 891	2 611	3
34	153	1 251	67	22 091	2 761	3
35	157	1 574	67	23 008	2 876	0
36	161	1 582	67	24 272	3 034	0
37	165	1 590	67	25 568	3 196	0
38	169	1 598	67	26 896	3 362	0
39	173	1 606	67	28 256	3 532	0
40	177	1 614	67	29 648	3 706	0

[a] All codewords shall be 8 bits in length.

Tabla 4.-Estándares para el código QR

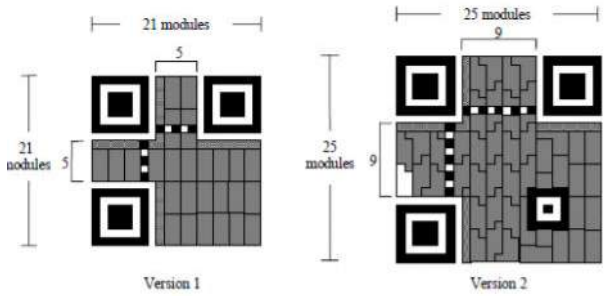

Figura 24.- Versión pequeña QR

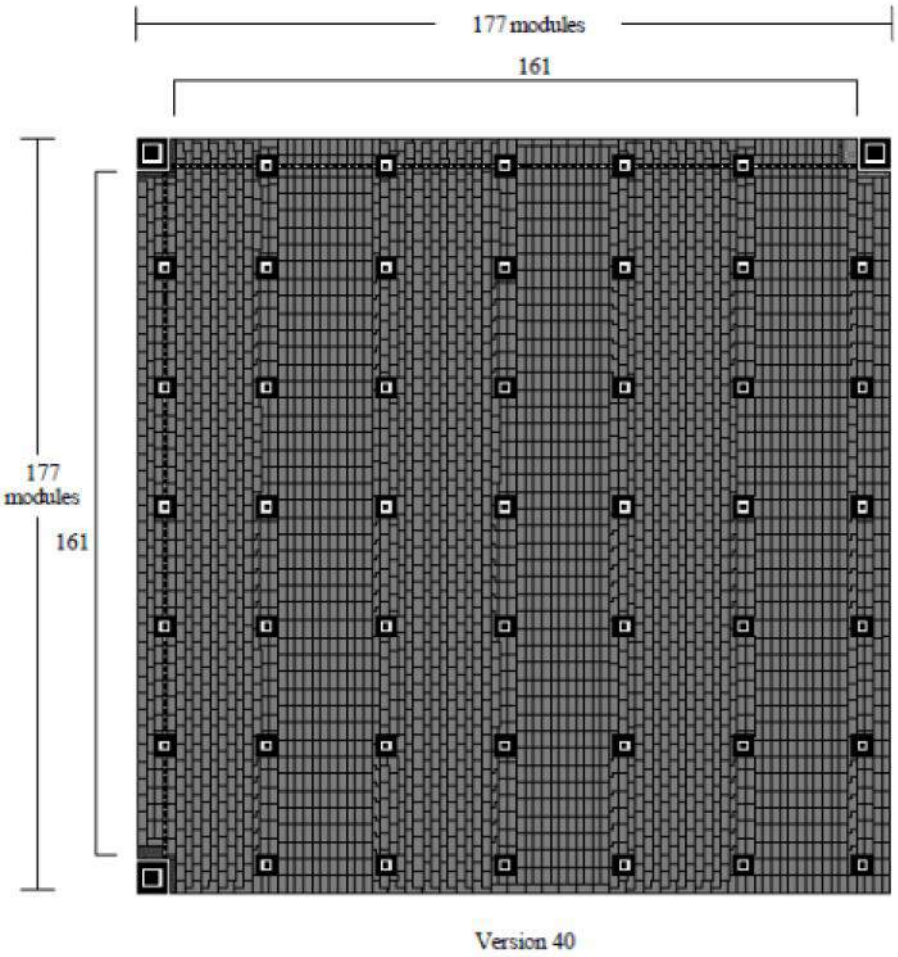

Figura 25.- Versión máxima QR

3.3 Codificación QR

Finder Pattern

A diferencia de los codigos de barras, los QR se puedenleer facilmente desde cualquier orientacion. Cuentan con patrones que sirven para determinar la posicion de la figura en tres de sus cuatro esquinas (ver figura 26). El camino que recorre una linea que cruza cada identificador de posicion, debe ser : 1:1:3:1:1 desde cualquier orientacion. Hecho esto para las tres esquinas de la figura, la determinacion de las dimensiones y orientacion es inmediata, permitiendo que la lectura de datos sea hasta 20 veces mas rapido que en otros sistemas matriciales.

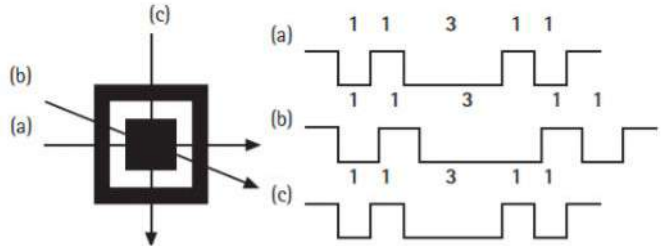

Figura 26.- Identificador de posición.

3.3.2 Separadores

Colocado entre cada detector de posición y región de datos, se coloca un separador que consiste en areas blancas de unmodulo de ancho; como se ve en la figura 27.

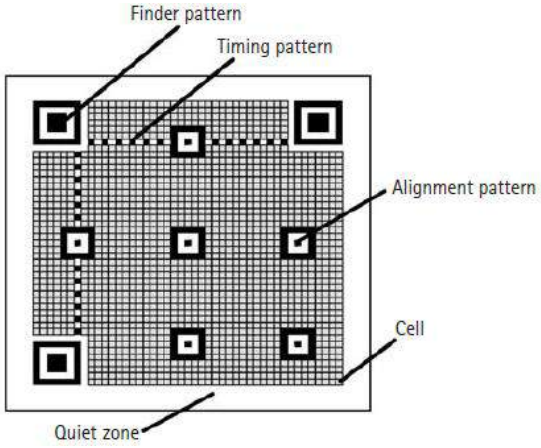

Figura 27.- Patrones de código QR

3.3.3 Patrones requeridos (timing pattern)

De acuerdo a la figura 27, los patrones horizontales y verticales,respectivamente consisten en un renglon o columna de un modulo de ancho, que tiene los colores alternados, comenzando y terminando con un bloque obscuro. El patron horizontal se encuentra a la altura del renglon 6 de la figura entre los separadores de los patrones de posicion superior.El patron vertical, de forma

analoga, se encuentra en la columna 6 de la figura, entre los separadores de los patrones de posicion izquierda. Estos patrones sirven para establecer la densidad del simbolo, y las coordenadas de los modulos a decodificar.

3.3.4 Patrones de alineamiento

Figura 28.- Deformaciones comunes de la imagen de un símbolo QR

Adicionalmente, los codigos QR cuentan con patrones de alineamiento, colocados a intervalos regulares. De esta forma, se compensan los errores inducidos por imágenes distorcionadas , ya sea por desalineacion camara-superficie, o lentes. Dos ejemplos de imágenes deformadas se pueden apreciar en la figura 28.

3.3.5 Área de datos

El area de color grisen la figura 27, esta destinada a los datos codificados. Es el resultado del proceso de codificar los caracteres en código binario, agregar los caracteres de relleno y dividir la secuencia entre el numero apropiado de bloques según las especificaciónes elegidas, así como la aplicación del patrón de enmascaramiento.

3.3.6 Quiet zone

Es un margen necesario para la corrrecta identificacion del simbolo. Generalmente se deja sin color. Debe tener como mínimo la anchura de cinco modulos , com se puede apreciar en las figuras 20 y 27, principalmente.

3.4 Propiedades de la codificación QR

La codificacion QR tiene cuatro niveles de correccion para datos perdidos (7, 15, 25 y 30%). El nivel de correccion se configura al crear la imagen. Utiliza un criterio llamado”Reed- Solomon”. Ver figura 28;

Figura 29.-Posibles pérdidas de datos que pueden ser recuperados por los algoritmos de corrección de errores

Los símbolos QR también pueden dividirse en partes (16 como máximo). Cada parte contiene información sobre el número de particiones en las que se ha dividido el original, y de cuál de todas las partes se trata,es decir; identificación individual.Ver figura 30.

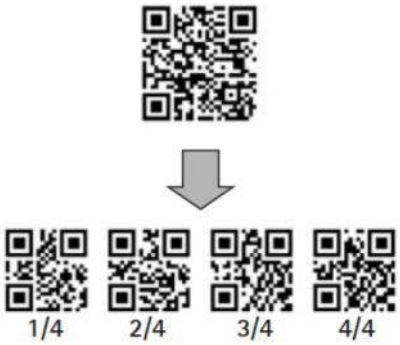

Figura 30.- Sub-símbolos QR. Pueden ser hasta 16 como máximo

3.5Proceso de codificación

El proceso de codificación consta de los siguientes pasos:

a) Analizarlos datos de entrada para identificar la variedad de datos a codificar. La codificación QR soporta diferentes modos para tratar los distintos tipos de caracteres de entrada. Se prueba entre esos modos hasta encontrar la conversión más eficiente de los datos a cadenas binarias. Se selecciona el nivel de error. Si el usuario no especifica la versión del símbolo (dimensión), el sistema automáticamente seleccionará el mínimo que pueda contener los datos a codificar (ver tabla 1).
b) Convertir los caracteres a un arreglo binario (*array*). Especificar los modos de codificación al inicio de cada segmento con formato distinto. Anadir el carácter de paro al final de la secuencia de datos. Separar el *array* resultante en palabras de 8 bits. Agregar los caracteres apropiados de relleno, a fin de completar todos los espacios disponibles que permita la versión del símbolo.
c) Dividir la secuencia en el número apropiado de bloques, de manera que sea compatible con el código de corrección de error elegido. Elegir dicho código para cada bloque de datos, así como registrar dicha secuencia al final del *array* binario.
d) Intercalar los bloques de datos con su correspondiente clave del código de corrección de errores elegido.
e) Colocar los módulos binarios en la matriz con los patrones de posición, separadores, patrones requeridos y alineamiento.

f) Aplicar los patrones de enmascaramiento a la región de datos del símbolo. Evaluar los resultados y seleccionar el enmascaramiento más eficiente en cuanto a que genere la menor cantidad de secuencias que puedan ser confundidas con algún patrón del símbolo. Ver figura 26.

La forma de aplicar la máscara, es implementar la operación XOR entre los datos y el patrón llamado mascara.

En la figura 31, se muestran los ocho tipos de enmascaramiento, así como sus códigos y algoritmo.

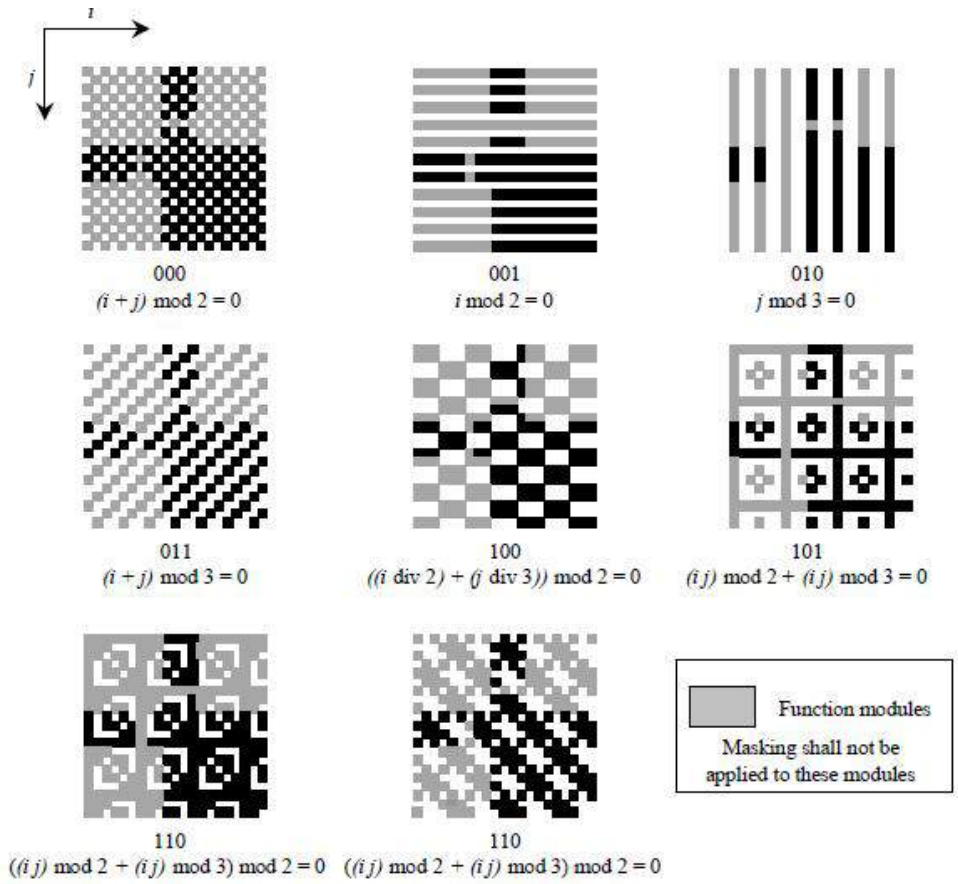

Figura 31.-Patrones de enmascaramiento

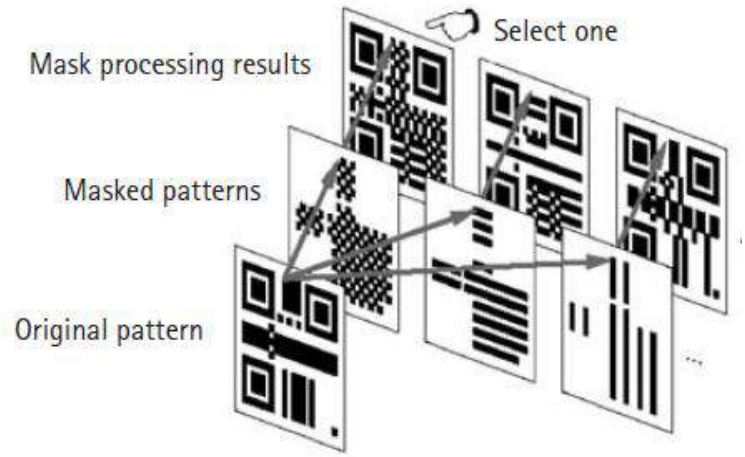

Figura 32.-Proceso de enmascaramiento

Los pasos para la lectura de los datos almacenados en un símbolo, son el inverso del proceso de la codificación.

a) Localizar y obtener la imagen del símbolo. Reconocer los módulos claros y obscuros como arreglos de “unos” y “ceros”.
b) Leer la información de formato.
c) Determinar la versión del símbolo.
d) Eliminar la máscara por medio de la operación XOR, entre la información plasmada y la máscara registrada en el símbolo.
e) Leer los caracteres de acuerdo a las reglas de posicionamiento para el modelo y determinar el código de corrección de errores.
f) Ejecutar la identificación de errores y corregirlos.
g) Mostrar el resultado.

El proceso se muestra en forma gráfica en la figura 33.

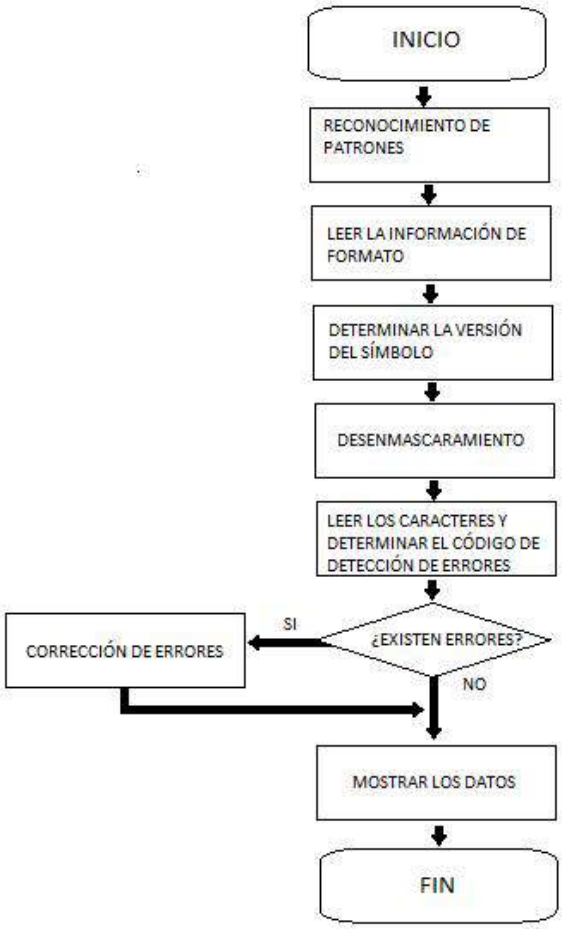

Figura 33.- Proceso de lectura de un símbolo QR

3.6Aplicaciones

Los códigos QR, por su capacidad de codificar información de varios tipos, han sido usados para publicar claves de acceso en aplicaciones, a páginas web o información promocional y de productos. Es muy común encontrar estos códigos en revistas o publicaciones, y tienen por objetivo que se utilice un dispositivo móvil para leer el código y así instalar o acceder a aplicaciones o promociones. Lo único que se tiene que hacer es usar una aplicación en el dispositivo que pueda leer códigos QR, apuntar la cámara al código a leer y dejar que el dispositivo haga lo propio. Así mismo, se están explorando formas de utilizarlo en la educación. Por ejemplo: codificar respuestas a cuestionarios, acceso a audio especializado para estudiantes de idiomas, etc. [15]

Capítulo 4

Tecnología RFID

4.0 INTRODUCCION

La tecnología de identificación por radiofrecuencia está teniendo una gran variedad de aplicaciones en distintos sectores de la actividad humana. Y el sector de la robótica no debe ser la excepción. Siendo parte tan importante del desarrollo de éste trabajo, es necesario exponer los principios de funcionamiento de éste sistema de identificación.

A pesar de parecer algo relativamente nuevo, la identificación por radiofrecuencia, tiene mucho tiempo de existir, por lo que, después de una breve reseña histórica, se describen los componentes del sistema seguido de las especificaciones técnicas y sus variedades.

Finalmente, se cubren los procedimientos para las transferencias de datos, distribución de la memoria, y aplicaciones de ésta tecnología, que van desde lavanderías, hasta implantes subcutáneos.

4.1Definición y Aplicaciones

Con el término RFID (*Radio Frequency Identificación*) se denomina a los sistemas de identificación automática remota de objetos mediante ondas de radio.

Los percusores de los sistemas RFID fueron los sistemas de detección automática de objetos. Una de las primeras patentes de estos sistemas fue un sistema radio-transmisor para detección de objetos diseñando por John Logie Baird en 1926 [16]. Después Robert Watson-Watt, en 1935 patentó el sistema: Detección y medición de distancias por radio ("Radio Detection and Ranging"-RADAR-).

La tecnología de comunicación pasiva, muy comúnmente usada en los actuales sistemas RFID, fue presentada por Henry Stockman en su trabajo: "Comunicación por energía reflejada" en 1948 [17].

Una de las primeras aplicaciones del sistema RFID tuvo lugar en la Segunda Guerra Mundial por los británicos en su sistema "*Identify Friend or Foe*" (IFF) desarrollado por la RAF(*Royal Air Force*) con la finalidad de distinguir aviones enemigos de los propios y así evitar bajas por "fuego amigo"[18].

En la actualidad, los sistemas RFID comerciales consisten en pequeños *transponders*, llamados etiquetas, tarjetas, o *Tag´s*, montados en objetos físicos; y un lector. El propósito fundamental de esta tecnología es transmitir la identidad de un objeto (similar a un número de serie) a un sistema de lectura, como el mostrado en la figura 34. Por tanto, el sistema RFID es un tipo de identificación automática similar a los códigos de barras ópticos [18].

Los *Tag´s* son dispositivos pequeños, encapsulados en objetos tales como:

- una tarjeta de identificación,
- un llavero,
- pulsera,
- una calcomanía,
- un botón, etc.

Y pueden verse en la figura 35

Figura 34.- Dispositivo Lector de Tag´s

Figura 35.- Diferentes versiones de Tag´s

Contienen antenas para permitirles recibir y mandar información vía radio-frecuencia. La principal ventaja sobre otros sistemas es que no requiere encontrarse en línea de vista con el lector para extraer la información. Pueden ser adheridas a un producto, un animal o una persona, como en la figura 36, bajo la función de monitores o registros.

Figura 36.- Ejemplo de aplicación de RFID

Los sistemas RFID se clasifican dependiendo del rango de frecuencias que usan. Existen cuatro tipos: [19]

- Frecuencia baja (entre 125 ó 134,2 kHz);
  - Control de acceso
  - Identificación de animales
  - Control antirrobo en autos
- Alta frecuencia (13,56 MHz);
  - Control de acceso
  - Bibliotecas y control de documentación
  - Pago en medios de trasporte
  - Control de equipaje en aviones.
- UHF o de frecuencia ultra elevada (868 a 956 MHz).
  - Cadenas de suministro.
  - Trazabilidad de objetos de valor.
  - Control anti falsificación.
  - Automatización de tareas de inventariado.

Pago de peaje en autopistas.

- Microondas (2,45 GHz).
  Pago de peaje en autopistas.
  Rastreo de vehículos.

Los sistemas UHF no pueden ser utilizados en todo el mundo porque no existe un estándar global para su uso.
Las etiquetas RFID pueden ser activas, semi-pasivas o pasivas. Las etiquetas pasivas no requieren ninguna fuente de alimentación interna y sólo se activan cuando un lector se encuentra cerca para suministrarles la energía necesaria (Figuras 38 y 39). Los otros dos tipos usan alimentación, típicamente una pila pequeña [19]. (Figura 37).

Figura 37.- Ejemplo de etiqueta activa

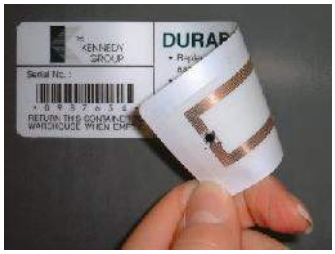

Figura 38.- Ejemplo de etiqueta pasiva

4.2 Etiquetas pasivas

Las etiquetas pasivas no poseen alimentación eléctrica. La señal que les llega de los lectores induce una corriente eléctrica pequeña y suficiente gracias al embobinado embebido (Figura 39). Esta energía es suficiente para operar el circuito integrado CMOS de la etiqueta, de forma que puede generar y transmitir una respuesta.

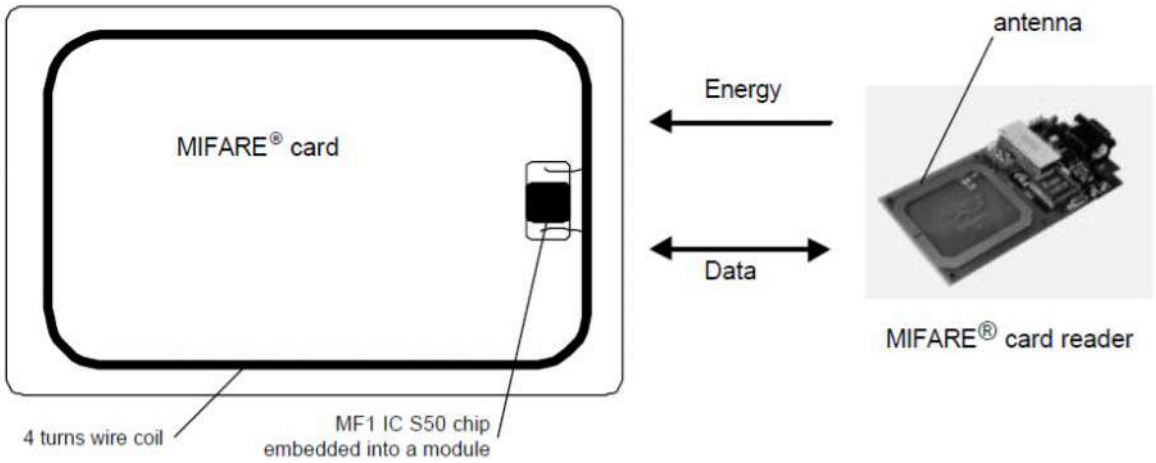

Figura 39.- Funcionamiento de una tarjeta pasiva

Las etiquetas pasivas suelen tener distancias de uso práctico comprendidas entre los 10 cm y llegando hasta unos pocos metros, según la frecuencia de funcionamiento, el diseño y tamaño de

la antena. Por su sencillez conceptual, son obtenibles por medio de un proceso de impresión de las antenas. Como no precisan de alimentación energética, el dispositivo puede resultar muy pequeño: pueden incluirse en una calca o insertarse bajo la piel.

4.3 Etiquetas activas

A diferencia de las etiquetas pasivas, las activas poseen su propia fuente de energía. Estas son mucho más fiables que las pasivas debido a su capacidad de establecer sesiones con el lector. Gracias a su fuente de energía son capaces de transmitir señales más potentes que las de las pasivas, lo que les lleva a ser más eficientes en entornos dificultosos para la radiofrecuencia como el agua (incluyendo humanos y ganado, formados en su mayoría por agua), y metal. También son efectivas a distancias mayores pudiendo generar respuestas claras a partir de recepciones débiles (al contrario que las pasivas). Por el contrario, suelen ser mayores y más caras, y su vida útil es en general mucho más corta. Ver figura 40.

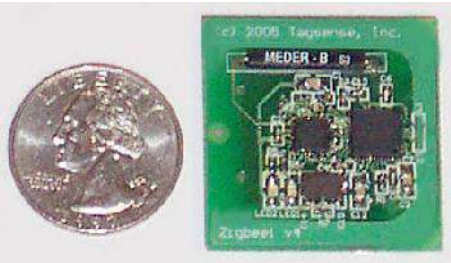

Figura 40.- Otra tarjeta activa

Actualmente, las etiquetas activas más pequeñas como la de la figura 40, tienen un tamaño aproximado de una moneda. Muchas etiquetas activas tienen rangos prácticos de diez metros.

4.4 Principios de la comunicación.

Los comandos son iniciados por el módulo de Lectura/Escritura (RDW) y controlados por la unidad de control de la tarjeta MF IC S50 de acuerdo a las condiciones de acceso válidas para el sector correspondiente [20].

1. *Request estandar \All*. Después de encenderse, la tarjeta responde a un comando de requisición enviado por el RWD a todas las tarjetas que se encuentren en el campo de alcance de la señal.
2. *Anticollision loop*. En el ciclo de anticolisión se lee el número de serie de la tarjeta. Si hay varias tarjetas en el área de alcance, se distinguirán por ese número de serie y por ese medio ser seleccionadas. Las tarjetas que no son requeridas, permanecerán sin funcionar hasta que vuelva a ocurrir una requisición.
3. *Select Card*. Con este comando, el RWD selecciona una tarjeta para autenticarla y ejecutar operaciones de memoria. La tarjeta responde con un (08h), que determina el tipo de tarjeta seleccionada.
4. *Authentication*. Después de seleccionar la tarjeta, el RWD especifica la localidad de memoria a la que se desea acceder, y usa la clave correspondiente para los tres pasos del proceso de autenticación. Solo después de una exitosa autenticación, las operaciones de memoria quedan habilitadas.
5. Operaciones de memoria. Después de la autenticación, se pueden ejecutar las siguientes operaciones de memoria.

- Leer bloque
- Escribir bloque
- Decrementos
- Incrementos
- Restauraciones
- Transferencias

4.5 Organización de la memoria

La memoria EEPROM de los Tag´s, se encuentra dividida en 16 sectores con 4 bloques cada uno [20].

1. El bloque 0 del sector 0, está protegido contra escritura por el fabricante.
2. El bloque número 3 de cada sector contiene información de acceso al propio sector.
3. El proyecto solamente ocupa el bloque 1 del sector 0 en la tabla 5

		Byte Number within a Block															
Sector	**Block**	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**Description**
15	3	Key A						Access Bits				Key B						Sector Trailer 15
	2																	Data
	1																	Data
	0																	Data
14	3	Key A						Access Bits				Key B						Sector Trailer 14
	2																	Data
	1																	Data
	0																	Data
:	:																	
:	:																	
:	:																	
1	3	Key A						Access Bits				Key B						Sector Trailer 1
	2																	Data
	1																	Data
	0																	Data
0	3	Key A						Access Bits				Key B						Sector Trailer 0
	2																	Data
	1																	Data
	0																	Manufacturer Block

Tabla 5.-Organización de la memoria

4.6 Aplicaciones

Dependiendo de las frecuencias utilizadas en los sistemas RFID; el coste, el alcance y las aplicaciones son diferentes.

Los sistemas que emplean frecuencias bajas tienen igualmente costos bajos, pero también baja distancia de uso. Los que emplean frecuencias más altas proporcionan distancias mayores de lectura y velocidades de lectura más rápidas.

Así, las de baja frecuencia se utilizan comúnmente para la identificación de animales o como llave de automóviles con sistema antirrobo.

En el sector Ganadero, por ejemplo, su utilidad consiste en poder tener un monitoreo efectivo de los animales, por ejemplo; rastrear sus movimientos, si han contraído o no alguna enfermedad, cambios de propiedad, nacimiento y muerte, número de identificador del animal y del propietario, edad, sexo, vacunas, raza y cruza. Como consecuencia natural del uso de ésta tecnología, será muy fácil determinar los resultados de programas sanitarios, alimenticios, o de otro tipo aplicados a los animales. Sobre todo si se trata de efectos de mediano o largo plazo.

Las etiquetas RFID de alta frecuencia se utilizan en seguimiento de libros en bibliotecas, y control de acceso en edificios, seguimiento de equipaje en aerolíneas, seguimiento de artículos de ropa y últimamente en pacientes de centros hospitalarios para identificación y un seguimiento de su historia clínica. A éste respecto, se han desarrollado varios prototipos de implantes subcutáneos RFID, sin embargo, el éxito final en la implantación de esta tecnología está sujeto a la superación de una serie de obstáculos, entre los que es necesario destacar los aspectos de seguridad y privacidad [21]. Un uso extendido de las etiquetas de alta frecuencia como identificación personal, substituirá a las anteriores tarjetas de banda magnética. Las etiquetas RFID de microondas se utilizan en el control de acceso en vehículos.

Algunas autopistas, como por ejemplo El carril de Telepeaje IAVE en las autopistas de CAPUFE En México utilizan etiquetas RFID para recaudación con peaje electrónico. Las tarjetas son leídas mientras los vehículos pasan.

En la actualidad los costos del RFID textil se han reducido considerablemente. Éstos pueden ser insertados en las prendas de forma muy discreta, dentro de los dobladillos, termosellados o simplemente cosidos. Gracias a este producto se consigue la optimización de recursos humanos y la reducción de hasta un 50% en la pérdida, extravío o robo de las prendas. [22]

Capítulo 5

BLUETOOTH

5.0 INTRODUCCION

Otro gran componente que debe ser mencionado es el protocolo de comunicación Bluetooth. Una forma adecuada de abordar el tema, debe ser la más familiar. Este capítulo comienza familiarizando al lector con la gran variedad de aplicaciones que se ven todos los días, y que nos hacen la vida más cómoda.
Luego, se habla de la historia del desarrollo de éste protocolo de comunicación, que comienza en Suecia, con la compañía Ericsson. Incluyendo el origen del nombre particular. Como parte central del capítulo, se explica el funcionamiento, consideraciones teóricas y, finalmente, aplicaciones.

5.1 ESPECIFICACIONES

El término Bluetooth es el término comercial y popular del estándar de comunicación inalámbrica IEEE 802.15.1 [23]. La tecnología inalámbrica Bluetooth, cuyo símbolo se muestra en la figura 41, es una tecnología de ondas de radio de corto alcance (2.4 GHz de frecuencia) cuyo objetivo es el simplificar las comunicaciones entre dispositivos informáticos, como ordenadores móviles, teléfonos móviles y otros dispositivos de mano. También pretende simplificar la sincronización de datos entre los dispositivos y otros ordenadores.

Bluetooth cuenta con 3 diferentes clases.

1. Clase 1. Cuenta con un alcance de cobertura aproximado de 100m.
2. Clase 2. Cuenta con un alcance de cobertura aproximado de 10m.
3. Clase 3. Cuenta con un alcance reducido de 1m.

El protocolo se basa en un modo de operación maestro-esclavo, o arquitectura cliente-servidor
La conexión de dispositivos por el medio Bluetooth sigue un procedimiento relativamente complejo: 1. Modo pasivo. 2. Solicitud. 3. Paginación. 4. Descubrimiento. 5. Creación de un canal de comunicación. 6. Emparejamiento. 7. Utilización de la red [24].

Figura 41.- Símbolo del protocolo Bluetooth

Ésta tecnología permite comunicaciones, incluso a través de obstáculos, a distancias de hasta unos 10 metros. Esto resulta muy conveniente en aplicaciones como: reproducir archivos de música, desde un teléfono celular y de manera inalámbrica en un reproductor casero o estéreo amplificado. También sirve para crear una conexión a Internet inalámbrica desde una portátil usando un teléfono móvil, conectar dispositivos a una PC como impresoras, ratón o teclado. Otro

caso es el poder sincronizar libretas de direcciones, calendarios etc. en algún teléfono móvil, ordenador de sobremesa y portátil automáticamente y al mismo tiempo.

5.2 Historia

En 1994 la empresa sueca Ericsson inició un estudio para investigar la viabilidad de una interfaz vía radio, de bajo costo y consumo para la interconexión entre teléfonos móviles y otros accesorios con el objetivo de eliminar cables entre aparatos. El estudio partía de un largo proyecto que investigaba sobre unos multicomunicadores conectados a una red celular, hasta que se llegó a un enlace de radio de corto alcance llamado MC link. Con el avance del proyecto quedó claro que este tipo de enlace podía ser utilizado en un gran número de aplicaciones, pues poseía como ventaja principal el hecho de basarse en un chip de radio relativamente económico.
A principios de 1997, otros fabricantes de equipos portátiles despertaron su interés por el avance del proyecto MC link y para que el sistema tuviera éxito, un gran número de equipos debería estar formado con esta tecnología. Ello fue lo que originó a principios de 1998, la creación de un Grupo de Especial Interés en Bluetooth (SIG), formado por cinco promotores y que fueron Ericsson, Nokia, IBM, Toshiba e Intel [25].
La tecnología bluetooth se usa para comunicar dispositivos que estén próximos, y tomó su nombre del rey vikingo Harald Blåtand, bluetooth en inglés; en español, dientes azules y su origen, la lengua danesa, significa de tez oscura. Harald Blåtand fue rey de Dinamarca (958-986) y Noruega (970-986). Durante el siglo X luchó por la comunicación y entendimiento de las tribus danesa, noruega y sueca; consiguiendo la unión de Dinamarca y Noruega bajo su corona. El logotipo de la tecnología bluetooth es la unión de las runas de sus iniciales, H (Harald) y B (Blåtand) [26]. Se muestra en la figura 42.

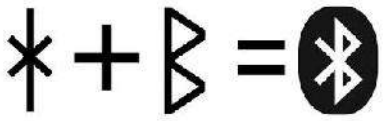

Figura 42.- Origen del símbolo Bluetooth

La especificación de Bluetooth define un canal de comunicación de máximo 720Kb/s (1Mbps de capacidad bruta).
La frecuencia de radio con la que trabaja está en el rango de 2.4 a 2.48 GHz con amplio espectro y saltos de frecuencia con posibilidad de transmitir en Full Dúplex con un máximo de 1600 saltos/s. Los saltos de frecuencia se dan entre un total de 79 frecuencias con intervalos de 1MHz; esto permite dar seguridad y robustez.

La potencia de salida para transmitir a una distancia máxima de 10 metros es de 0 dBm (1 mW), mientras que la versión de largo alcance transmite entre 20 y 30 dBm (entre 100 mW y 1 W).
Para lograr alcanzar el objetivo de bajo consumo y bajo costo, se ideó una solución que se puede implementar en un solo chip utilizando circuitos CMOS. De esta manera, se logró crear una solución de un área de 81mm^{2}y que consume aproximadamente 97% menos energía que un teléfono celular común [27].

El protocolo de banda base (canales simples por línea) combina conmutación de circuitos y paquetes. Para asegurar que los paquetes no lleguen fuera de orden, los slots pueden ser reservados por paquetes síncronos. Un salto diferente de señal es usado para cada paquete. Por

otro lado, la conmutación de circuitos puede ser asíncrona o síncrona. Tres canales de datos síncronos (voz), o un canal de datos síncrono y uno asíncrono, pueden ser soportados en un solo canal. Cada canal de voz puede soportar una tasa de transferencia de 64 Kb/s en cada sentido, la cual es suficientemente adecuada para la transmisión de voz. Un canal asíncrono puede transmitir como mucho 721 Kb/s en una dirección y 56 Kb/s en la dirección opuesta, sin embargo, para una conexión asíncrona es posible soportar 432,6 Kb/s en ambas direcciones si el enlace es "simétrico."

En Bluetooth todos los datos que se envían a través del canal son fragmentados y enviados en paquetes (ver figura 43). Además la información se encuentra protegida mediante códigos detectores y/o correctores de errores. En cada ranura solo se puede enviar un paquete. El receptor los recibirá y los procesará empezando por el bit menos significativo. El esquema se muestra en la figura 43.

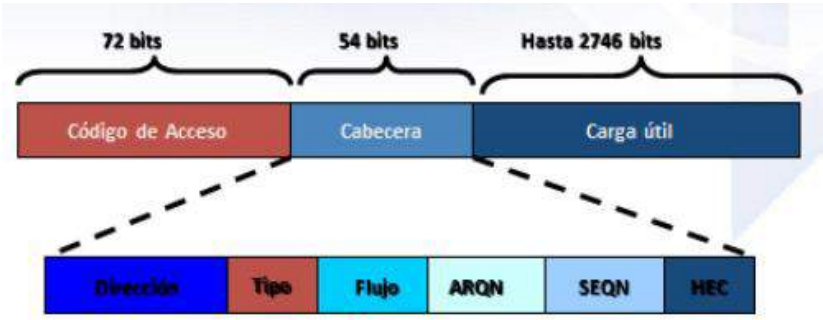

Figura 43.- Paquete de datos de la comunicación Bluetooth

5.3 Composición del paquete:

• Código de acceso (72 bits):
Es usado para sincronización, identificación y compensación.

• Cabecera (54 bits):
Contiene información del control de enlace con 6 campos:

1. Dirección o AM_ADDR: Dirección temporal de 3 bits que se utiliza para distinguir los dispositivos activos en una piconet , siendo la dirección 000 la dirección broadcast .

2. Tipo: Define qué tipo de paquete es enviado y cuántos slots va a ocupar.

3. Flujo o Flow: El bit de control de flujo es usado para notificar al emisor cuando el buffer del receptor está lleno y que debe de dejar de transmitir, en ese caso el bit tendrá el valor "0". \

4. ARQN: Bit de reconocimiento de paquetes recibidos paquetes correcto o incorrecto (último paquete recibido). Si es un "1" es un ACK, y con un"0" un NAK.

5. SEQN: Bit que se va invirtiendo para evitar retransmisiones en el receptor.

6. HEC: Código de redundancia para comprobar errores en la transmisión.

7. Campo de datos o carga útil (hasta 2746 bits): Contiene el conjunto de datos que supone la información a transmitir.

5.4 Usos que se le pueden dar a la tecnología Bluetooth

Una de las máximas ventajas de Bluetooth, no es solo el poder conectar dispositivos a ordenadores. Se puede usar para interconectar básicamente cualquier par de dispositivos si ambos utilizan esta tecnología.

Se podrán conectar diferentes accesorios y productos electrónicos, desde cámaras fotográficas a periféricos informáticos, sin ninguna conexión física entre ellos. Los teléfonos móviles y las PDAs se aprovechan de esta tecnología en muchos entornos. Se pueden transferir archivos de cualquier tipo. Se puede utilizar para realizar facturaciones en pequeñas empresas, aeropuertos, oficinas etc.

La diferencia entre Wireless y Bluetooth es que la primera tiene más alcance y potencia, aunque también sea más costosa de implementar. Se debería pensar en ambas como dos tecnologías que se pueden complementar.

Capítulo 6

RECTIFICACIÓN DE IMAGEN

INTRODUCCIÓN

Con el fin de poder deducir la posición de un objeto y la orientación de una imagen, se necesitan los detalles de la posición de la cámara y la orientación en el espacio relativa a algún sistema de coordenadas de referencia conocido como "Sistema de coordenadas del mundo". Además es preciso conocer la geometría de la cámara; a veces llamada "Geometría de la cámara", y algún método para encontrar los distintos parámetros presentes en el modelo. Éste proceso es conocido como "Calibración de la cámara".

La elección del modelo de la cámara depende de la aplicación particular del sistema de visión. En sistemas de más de una cámara, los sistemas de coordenadas asociados con cada cámara y el sistema de coordenadas del mundo probablemente estarán desalineados, mientras en los sistemas de una sola cámara se pueden elegir los sistemas de coordenadas de forma que el sistema del mundo y de la cámara tengan los ejes alineados.

Además de describir la geometría de la cámara, el modelo de la cámara también describirá varias características internas de la misma, tales como longitud focal y distorsión de las lentes [29].

En el presente capítulo se definen algunas de las distorsiones de imagen y las consideraciones teóricas del proceso de calibración de la cámara, cuyo resultado arrojará una rectificación de imagen.

6.1 Deformaciones

Puede ser común esperar que la imagen obtenida de una cámara sea una fiel reproducción de la realidad. Pero no es así. Existen deformaciones en la imagen producto de factores extrínsecos e intrínsecos al dispositivo.

Como factor intrínseco se tiene la distorsión geométrica. Consiste en que "Las líneas rectas, no se ven rectas".

Este efecto puede manifestarse en tres formas distintas:

- Distorsión de barril
- Distorsión de cojín
- Distorsión de mostacho

Figura 44.- Deformaciones esféricas

Las distorsiones se ejemplifican en la figura 44. La distorsión de barril es típica de lentes de distancia focal corta, mientras que la distorsión de cojín se presenta en lentes con distancia focal grande como los teleobjetivos [28].
En la deformación por perspectiva, algunas propiedades geométricas se conservan, tales como colinealidad (una línea recta se proyecta en una recta), mientras otras propiedades no, por ejemplo paralelismo, en general las líneas paralelas no se ven paralelas en la imagen.
Ésta deformación es independiente del lente y depende de la distancia al objeto.
La distorsión geométrica no debe confundirse con la deformación por perspectiva, que se muestra en la figura 45.
En dicha figura (Figura 45), se tiene el mismo objeto, pero se observa la deformación conforme se disminuye la distancia al mismo.

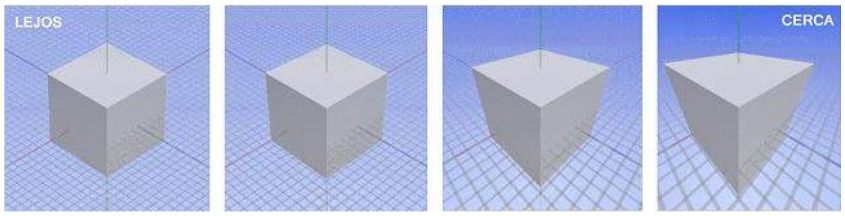

Figura 45.- Deformación por perspectiva

6.2 Parámetros de la cámara

Debido a la relativa lejanía de la webcam1 de la mesa de trabajo, así como a su posicionamiento centrado, la deformación por perspectiva prácticamente desaparece. Por lo que el proceso corresponde solamente a la corrección de la deformación geométrica tipo “distorsión de barril”.

Para corregir la imagen, es necesario reubicar los pixeles en la imagen. Es decir; se vuelve necesario conocer a qué punto del CCD de la cámara corresponde un punto en la imagen del objeto.

La formulación matemática, principalmente, trata de relacionar linealmente las coordenadas tridimensionales del punto en el mundo real (X, Y, Z) con sus coordenadas 2D en el plano imagen (x, y).
La conversión geométrica del objeto tridimensional en un objeto 2D en el plano imagen de la cámara obedece las leyes de la proyección perspectiva o proyección central, en la que los rayos de luz procedentes del objeto pasan a través del centro de proyección o centro óptico de la cámara y se plasman en la película del sensor. Esta proyección es el modelo de cámara más básico llamado “pinhole”, o cámara obscura. Dicho modelo fue el primer dispositivo fotográfico. Se muestra en la figura 46.

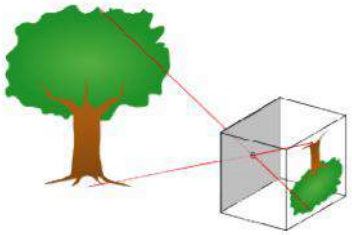

Figura 46.-Ejemplificación de una cámara estenopeica o “pinhole”. Los rayos de luz provenientes de un objeto atraviesan un pequeño agujero para formar la imagen

Si los componentes ópticos de la cámara fueran perfectos, la transformación entre la imagen bidimensional y el objeto tridimensional en el espacio sería perfectamente lineal y fácil de resolver. Pero los lentes del sensor producen distorsiones que no son lineales y que afectan a la precisión de la transformación. La distorsión radial desplaza los puntos de la imagen radialmente a partir del centro, mientras que la tangencial o descentrada los desplaza perpendicularmente a la línea radial. La causa de la primera es el incorrecto pulido de la lente, mientras que el de la segunda es la falta de alineación entre los componentes ópticos.
Aunque en la práctica, la única distorsión que produce verdaderos problemas es la radial [29].

El modelo pinhole consiste en un centro óptico C, en donde convergen todos los rayos de la proyección, y un plano de imagen XY, en el cual la imagen es proyectada. El plano de la imagen está ubicado a una distancia focal f del centro óptico y perpendicular al eje óptico Z. En la figura 47 se muestra el modelado de dicho arreglo.

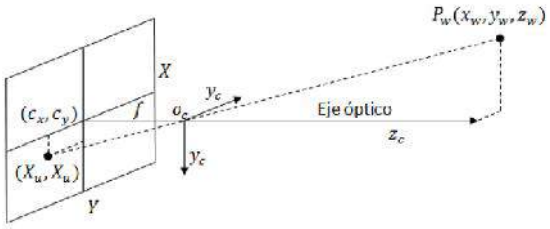

Figura 47.-Geometría del modelo de proyección de perspectiva

En dicha figura, el punto $P_w = (x_w, y_w, z_w)$ son las coordenadas 3D del punto. Mientras que $P_c = (x_c, y_c, z_c)$ son las coordenadas 2D del punto en el sistema de coordenadas de la cámara.
La forma de relacionar éstos puntos es mediante la trasformación

$$P_c = \mathbf{R}P_w + \mathbf{t} \qquad \text{ec. 20}$$

Donde **R** es la matriz de rotación y **t** es un vector de translación.
La transformación entre el sistema de coordenadas tridimensional de la cámara y una imagen del punto está dado por: (X_u, Y_u) donde por triángulos semejantes:

$$X_u = fx_w/z_w \qquad \text{ec. 21}$$

y

$$Y_u = fy_w/z_w. \qquad \text{ec. 22}$$

Considerando la distorsión radial, las coordenadas se presentan por

$$X_d + D_x = X_u \qquad \text{ec.23}$$

y

$$Y_d + D_y = Y_u. \qquad \text{ec. 24}$$

Donde

$$D_x = X_d(\delta_1 r^2 + \delta_2 r^4 + \cdots)$$

$$D_y = Y_d(\delta_1 r^2 + \delta_2 r^4 + \cdots)$$

$$r = (X_d{}^2 + Y_d{}^2)^{1/2}$$

ec. 25

Por tanto, (X_d, Y_d) son las coordenadas distorsionadas en el plano de la imagen [30].
Cabe mencionar que la series de la ec. 25 son infinitas. No obstante, siguiendo la experiencia demostrada en Tsai (1987) con un único coeficiente se obtienen buenos resultados [31].

CURVAS DE BÉZIER
Las curvas de Bézier se utilizan ampliamente en modelado de curvas suaves y superficies en 3D [32].
Para trazar una recta entre dos puntos basta con conocer sus posiciones.
Si en lugar de unir dos puntos con una recta se unen con una curva, surgen los elementos esenciales de una curva Bézier. Los puntos P_i se llaman puntos o vértices de control y el polígono determinado por ellos se llama *polígono de control de la curva de Bézier.* Véase figura 48.

Dependiendo del número de puntos utilizados, las curvas pueden ser: lineales (2puntos), cuadráticas (3 puntos), o cúbicas (4 puntos) como se ve en la figura 48.

Las curvas cuadráticas y cúbicas son muy corrientes. Las curvas de grados superiores son más difíciles de evaluar. Cuanto más complejas son las superficies que se necesitan, las curvas de bajo orden son menos apropiadas.

Los modernos sistemas de imágenes como PostScript, Asymptote y Metafont usan curvas de Bézier desdobladas, compuestas por curvas cúbicas de Bézier para dibujar las formas de las curvas [33].

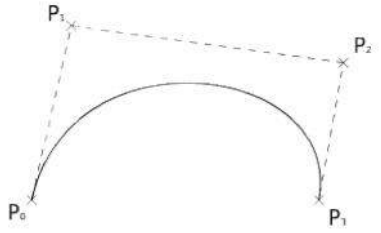

Figura 48.- Curva cúbica de Bézier

Dados n + 1 puntos en el plano, P0, · · ·, Pn, la ahora llamada curva de Bézier definida por ellos está dada por el algoritmo siguiente:

$$P(u) = \sum_{i=0}^{n} \binom{n}{i} (1-u)^{n-i} u^i \; P_i$$

ec. 26

Donde $\binom{n}{i} = \frac{n!}{i!(n-i)!}$ $0 \leq u \leq 1$

ec. 27

Capítulo 7

Implementación del sistema de Lectura/ Escritura con RFID

7.0 INTRODUCCION

Se describen los componentes del sistema de lectura del modo RFID. Éstos son:

- Lector-Grabador RFID
- Transmisor Bluetooth
- Receptor Bluetooth

Los tres componentes son de fácil acceso en el mercado. El lector- grabador y el transmisor se comunican con el exterior vía RS232.
El receptor Bluetooth es más fácil de implementar, ya que no necesita ser configurado.

7.1 Lector-grabador RFID

Para la lectura de las tarjetas RFID, se adquirió el modulo Lectura / Escritura RFID de 13.56MHz modelo **YHY502CTG,** marca Mifare® (figura 49), con antena integrada en el PCB. Éste dispositivo es denominado WRD por el fabricante.

Solamente es necesario que dicho módulo reciba las órdenes, como leer o escribir en la tarjeta. Dichas órdenes pueden provenir de una PC o un microcontrolador. Adicionalmente, no requiere que el usuario gestione los procesos de: “Request”, “Anticoll” o” Selection”, que son previos a la ejecución de operaciones (Ver capítulo 4). El modulo lo hace automáticamente.

Cuando éste módulo detecta un Tag dentro del campo de lectura, encenderá un LED rojo y el pin “SIG” cambiara su estado de “1” a “0” lógico; indicando el tipo de evento [34]. (Ver figura 51).

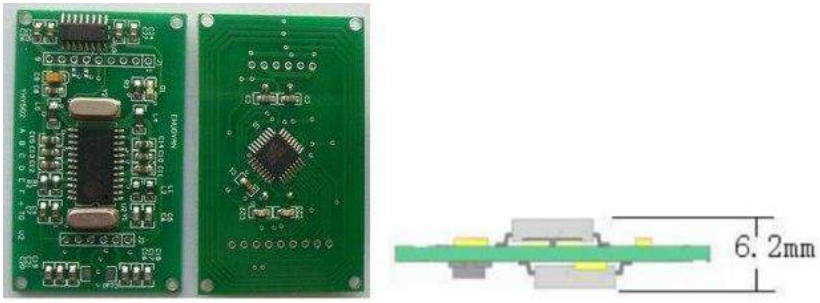

Figura 49.- Anverso, Reverso y perfil del módulo RWD

Figura 50.-Tag´s utilizados en el proyecto

El modulo está integrado con un microcontrolador de 8bits, procesador digital de señal analógico y digital, y los componentes asociados necesarios en una tarjeta PCB de doble cara [34]. (Ver figura 49).

Como es apreciable en la figura 49, se tienen perforaciones donde deberán soldarse las terminales eléctricas para su funcionamiento y comunicación. La distribución de pines, y un acercamiento al módulo se dan en la figura 51.

Los Tag´s que se utilizaron en el proyecto fueron: tarjeta, pulsera y llavero (Ver figura 50).

Pin	Symbol	IO Type	Description
J1-1	RXD	I	Uart Receiver
J1-2	TXD	O	Uart Transmitter
J1-3	OUT1	O	Output 1
J1-4	OUT2	O	Output 2
J1-5	RST	I	Reset, active-low, floating for power-on reset by default
J1-6	BUZ	O	high level drive, connect to buzzer drive circuit
J1-7	SIG	O	Interrupt output, LOW level indicates card in the field
J1-8	VCC	Power	Power positive
J1-9	GND	GND	Power Negative

Figura 51.- Configuración de pines del módulo RWD

7.1.1Protocolo de comunicación

La comunicación entre el Host (microcontrolador o PC) y el modulo, se da bajo el protocolo de comunicación serial RS232 a 19200 Baudios N, 8, 1. (19200 bits por segundo, sin bit de paridad, ocho bits de datos y un bit de paro) [34]. Primero el Host manda el comando, luego, el modulo ejecuta la operación requerida y envía una respuesta al host. De esta manera, el host recibe la información solicitada, además de un código extra que indica si la operación fue exitosa.

El formato de los las órdenes del host es el siguiente (Tabla 6):

Header	Length	Command	Data	CSUM
2 Byte	1 Byte	1 Byte	N Bytes	1 Byte

Tabla 6.-Formato de las órdenes del Host

- Header.- El encabezado es de 2 Bytes. Indica el comienzo de un bloque de información. Estos 2 Bytes deben ser *0xAA 0xBB.*
- Length.-Este byte indica la longitud de la información que se manda. Se incluye a sí mismo, Comandos y los Bytes de datos.
- Command.- Este Byte se usa para expresar la operación que se quiere ejecutar.
- Data.-Son parámetros asociados a la operación requerida en Command.
- Csum.- Este el Byte de Checksum. Se usa para verificar la validez de los datos obtenidos por el host. Se obtiene aplicando la operación XOR a todos los Bytes que conforman el paquete. Excepto el encabezado y el propio checksum. Es decir:
  CSUM=Length ⊕ Command ⊕ Data[0] ⊕ Data[1]… ⊕ Data[n-1].

El formato de los datos que recibe el host es el siguiente (Tabla 7):

Header	Length	Status	Response	CSUM
2 Byte	1 Byte	1 Byte	N Bytes	1 Byte

Tabla 7.-Formato de respuesta al Host

- Header.- El encabezado es de 2 Bytes. Indica el comienzo de un bloque de información. Estos 2 Bytes deben ser *0xAA 0xBB.*
- Length.-Este byte indica la longitud de la información que se manda. Se incluye así mismo, a los comandos y los Bytes de datos.
- Status .- Si la operación fue exitosa, regresa el comando previamente mandado por el host. En caso contrario, regresa el "complemento a uno" del comando.
- Response.- Contiene el resultado de alguna operación. Puede ser vacío.
- Csum.- Este el Byte de Checksum. Se usa para verificar la validez de los datos obtenidos. Se obtiene aplicando la operación XOR a todos los Bytes que conforman el paquete. Excepto el encabezado y el propio checksum.

7.2 HC-05

El modulo maestro HC-05 permite transmitir datos recibidos en formato RS-232 por vía bluetooth. También puede recibir datos vía bluetooth y mandarlos vía RS-232, como lo ilustra la figura 52. Se consigue comercialmente y viene listo para entrar en funcionamiento. La apariencia del módulo se muestra en la figura 53.

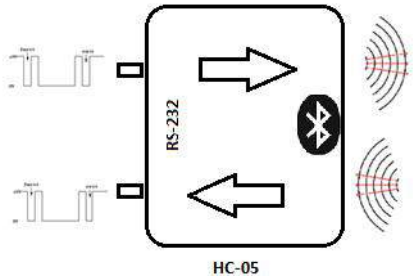

Figura 52.- Representación esquemática del módulo HC-05

Figura 53.-Anverso y reverso del módulo comercial HC-05

El dispositivo viene configurado de fábrica para comunicarse a una velocidad de 9600 Baudios, sin bit de paridad y sin control de flujo [36]. Con el fin de ajustar a los Baudios necesarios para la comunicación con el módulo RWD (115200 baudios), es necesario configurarlo.

Dicha configuración consiste en mandarle comandos desde una computadora como lo sugiere la figura 56. De ahí se ve que, los pines 31 y 32 del módulo HC-05 llevan conectados dos LED's a través de dos resistencias de 470 Ω para indicar los modos de funcionamiento de este.

- LED conectado al PIN31: parpadeo lento (1 Hz) indicación de que el módulo está en modo comandos AT. Parpadeo rápido (2 Hz) indica modo comunicación intentando vincularse a otro dispositivo. Doble parpadeo por segundo indica de que el dispositivo se ha vinculado y está listo para comunicarse.
- LED conectado al PIN32: antes de vincularse esta apagado, después de la sincronización esta encendido.

La configuración se realiza a través del envío de comandos. Con ese fin se utiliza la interfaz de distribución libre "Terminal V1.9b" (Ver figura 55)

La comunicación con la computadora, se hace vía RS-232. Los niveles de voltaje que se manejan en un puerto serial de conector DB-9 en las computadoras, no son los mismos que los que manejan los dispositivos y microcontroladores. Éstos utilizan niveles de 0 a 5V para CERO lógico y UNO lógico, respectivamente (Niveles TTL). Mientras que la computadora utiliza en su puerto de salida +15V y -15V para CERO lógico y UNO lógico, respectivamente [35]. Ver figura 54.

El acoplamiento de niveles de voltaje se realiza utilizando el circuito integrado MAX232 como se muestra en el diagrama de la figura 56.

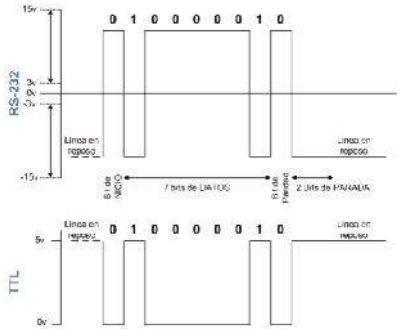

Figura 54.- Ejemplificación de señales de comunicación serial

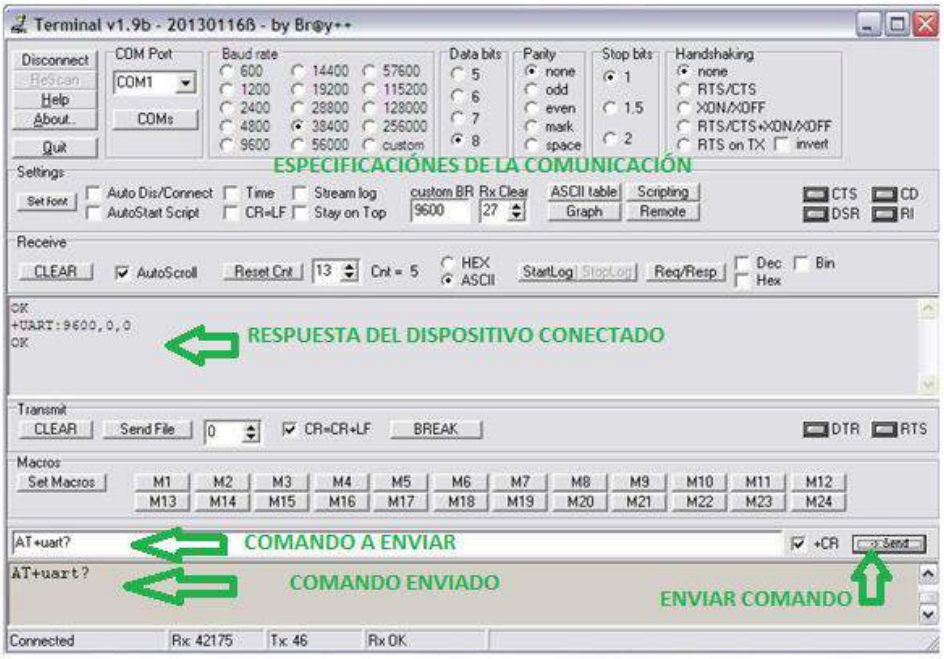

Figura 55.- pantalla de la aplicación "Terminal " para configurar el HC-05

Para especificarle al dispositivo que los datos que recibirá, son comandos de programación; en lugar de información para transmitir, se tienen que seguir los siguientes pasos: [36].

- Conectar la terminal KEY (PIO11) del HC05 a nivel “alto”.
- Encender el modulo. Esto lo pondrá en modo de comandos.
- Usando el mencionado software “Terminal”, a una tasa de Baudios de 38400, enviar el comando “AT+ROLE=1\r\n”. si la comunicación fue exitosa, el modulo enviara de regreso “OK\r\n”. Se da un ejemplo del uso del programa en la figura 50.
- Modificar los parámetros deseados. Ver apéndice.
- Conectar el pin PIO11 del HC05 nuevamente a nivel bajo. Reiniciar.

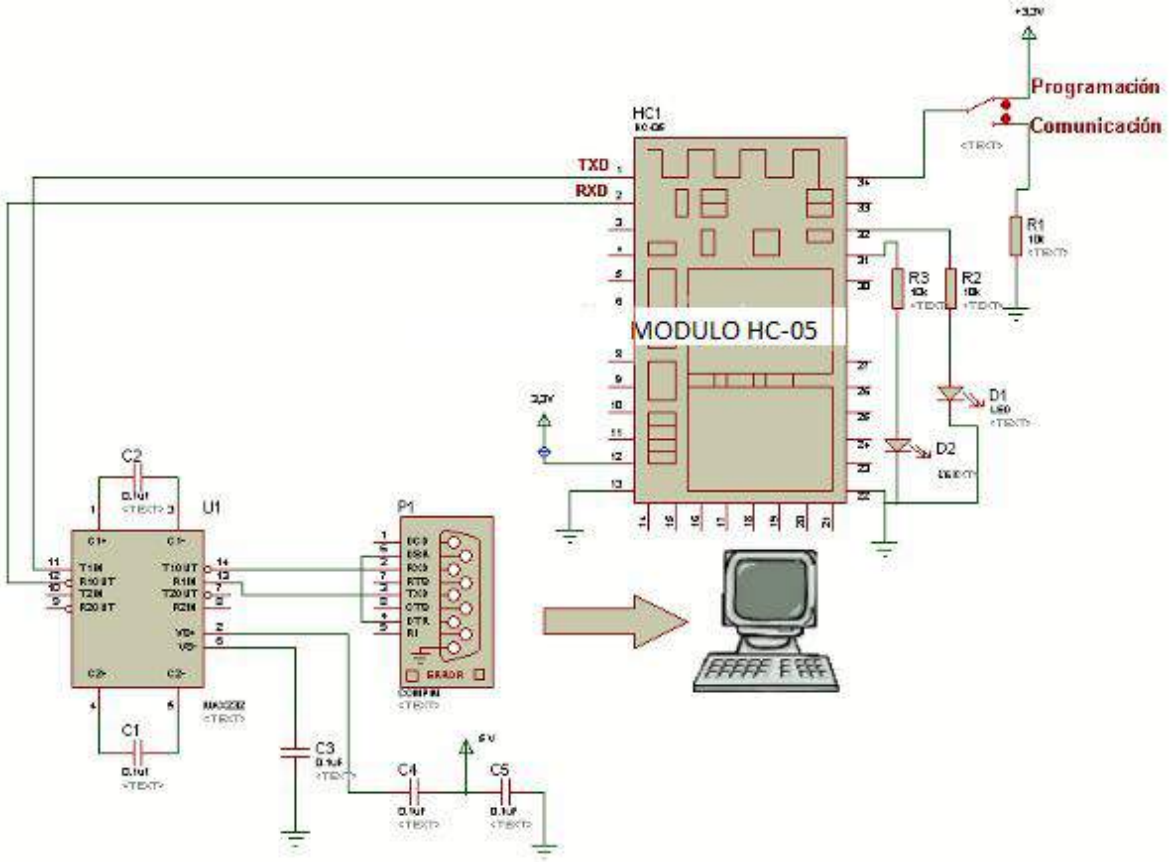

Figura 56.-Esquemático para realizar la configuración del módulo HC-05

Nota: Muchas de las conexiones que aparecen en la figura, asociadas al HC-05, ya no son necesarias, ya que el modulo, se vende sobre una plataforma de PCB con regulador de voltaje y componentes asociados.

La lista completa de comandos para éste dispositivo se muestra en el apéndice 6.

7.3 ADAPTADOR BLUETOOTH-USB

Una vez configurado y puesto en marcha el dispositivo emisor, es necesario que sea reconocido por la computadora. En caso de que ésta no cuente con comunicación bluetooth integrada, se puede adquirir un adaptador con conexión USB mostrado en la figura 57.
Su tamaño y costo son bastante reducidos (Tabla 8), no necesita un driver específico, y resulto tener un perfecto desempeño durante el desarrollo del presente proyecto.

Muchos de estos módulos poseen la misma dirección Bluetooth. Si se piensa utilizar para conectar una computadora a un celular u otro dispositivo funcionará perfectamente. Sin embargo si se conectan varios de estos módulos a una computadora tal vez el equipo de cómputo no pueda controlarlos todos al mismo tiempo

Figura 57.- Modulo adaptador de Bluetooth USB

Medidas	
Altura	**21 mm**
Ancho	**17mm**
Fondo	**6mm**
Peso	**2,1 g**

Tabla 8.- Medidas del módulo adaptador.

Sus especificaciones se mustran en la tabla 9:

Características	
Interface	**USB Ver 2.0 de alta velocidad**
Sistemas soportados	Windows98/98SE/ME/2000/XP/Vista/Win7
Alcance	de 25 a 100m, esta distancia puede variar por paredes, señales de radio, etc.
Alimentación	Vía USB, con voltaje de 5V
Tasa de transferencia	superior a los 3Mbps
Banda de frecuencia	de 2,44Ghz a 2,483 GHz
Soporta multilenguaje	**Si**
“Plug & play”	**Si,** No necesita drivers

Tabla 9.- Especificaciones de el adaptador

Capítulo 8

PROCEDIMIENTO

INTRODUCCIÓN

Este capítulo trata sobre el desarrollo del proyecto. Se integran ahora las nociones sobre los módulos descritos en los capítulos anteriores.
Trata primeramente del propósito del trabajo, las características que le da su entorno de trabajo. Se revisan las ventajas en comparación con otros sistemas de reconocimiento de objetos, y se da una idea general del funcionamiento del sistema.
A continuación se describe la forma de interactuar de los componentes sin entrar en demasiados detalles técnicos para una clara comprensión.

Luego se describen los componentes de forma individual. Solamente la unidad RFID se describe a detalle, ya que fue la única construida por el autor. Dichos componentes son:

- Unidad RFID,
- Lectura QR,
- Computadora,
- Interfaz USB
- Robot KUKA.

Luego se describe cómo están codificados los datos de los Tag´s.
La parte de proceso es donde se encuentra mayormente la descripción de las consideraciones técnicas y se divide en:

- Criterios de funcionamiento
- MATLAB
- Bluetooth
- QR
- Manejo de la información

Para que el robot realice todo lo antes descrito, fue necesario crear varias funciones dentro de lo que fue el programa hecho en MATLAB y también se cubren a detalle. Una vez resueltos los detalles técnicos, se describe la interfaz del programa de control.

Finalmente, se proponen aplicaciones donde un proyecto como éste puede aplicarse con mucha utilidad. Éstas incluyen, Industria, Salud y Educación.

8.1 Propósitos

El presente proyecto demostró y cumplió el propósito de integrar en un sistema,

- Tecnologías recientes de recuperación de información.
- Visión artificial
- Brazo robótico industrial
- Transmisión inalámbrica de datos

El sistema debe ser capaz de identificar objetos específicos, leer su información, y tomar una decisión conjuntando dicha información con el resultado del procesamiento de imágenes.

8.2 Entorno de Trabajo

El proyecto se desarrolla en un ambiente de iluminación controlada de laboratorio. Se simula la llegada de material a un lugar fijo destinado a la identificación de objetos. De ahí, el brazo robot tomará las piezas y las colocará en otra superficie llamada en adelante "mesa de trabajo". Sobre dicha mesa de trabajo se representan acciones que puede ejecutar el brazo robot.
Aunque en el presente trabajo se analizará un ejemplo de laboratorio, la aplicación puede extenderse sin problemas a una aplicación industrial, pues se desarrolló un protocolo de análisis de tal manera que insertando las consideraciones de la industria, la aplicación podría funcionar apropiadamente en la selección de suministros que lleguen a una planta, en la separación de material diverso almacenado, en el transporte automatizado de material dentro de la nave industrial

En caso de llevarse a una nave industrial, habrá variación en la iluminación incluso a lo largo de la jornada de trabajo, por lo que habrá que hacer algunos ajustes en cuanto a los niveles de binarización de imagen. Esto afecta tanto a la imagen de retroalimentación del sistema (webcam 1), como a la lectura de códigos QR. Estos inconvenientes no afectarán a las lecturas RFID. Además, usando RFID, la transmisión de datos a la PC es inalámbrica. Lo cual contribuye a un ambiente de trabajo más seguro y agradable.

8.3 Características y ventajas

Existen varios trabajos sobre identificación de piezas por visión artificial, sin embargo, requieren de una gran potencia de procesamiento. Esto debido a que, a partir de la imagen obtenida, se busca el patrón de identificación. Es necesario desarrollar barridos en toda la imagen, eso sin tener en cuenta que la pieza puede estar un poco movida. Buscar varios patrones para la identificación de tan solo una figura, puede incrementar de forma estrepitosa el costo computacional.
Además, la gran mayoría de aplicaciones parecidas no hacen previsión de que el destino de la pieza se pudiera encontrar ocupado.
Una ventaja de la presente propuesta es que evita los inconvenientes computacionales antes mencionados. La diferenciación de piezas no se hace a partir de su apariencia, sino gracias a elementos de identificación colocados en éstas. De ésta manera, no solamente se hace el reconocimiento más rápidamente, también se tiene acceso a datos adicionales. Los elementos de identificación pueden guardar una gran cantidad de información. Por ejemplo: pueden guardarse fechas de elaboración y caducidad, línea de producción, turno, lote, registro de tiempos y horarios, etapas del proceso, dimensiones, peso, numero de parte, color, acabados, destino, datos para almacenaje, manipulación y distribución.
Obtener, o grabar toda esa información de un producto con el menor costo computacional puede tener grandes ventajas. Como son:

- Menor tiempo de procesamiento de información. (El mismo ordenador puede gestionar algún otro proceso al mismo tiempo).
- Oportunidad de realizar actualizaciones en la información de un producto durante el proceso sin interrupciones.
- Mayor velocidad en la aplicación.

El obtener la información completa de una pieza defectuosa puede ser de gran utilidad, ya que los datos guardados en su identificador que estén involucrados con las etapas del proceso arrojarán una base estadística sólida para determinar las causas de la falla.
Los identificadores son de dos tipos: códigos QR y Tag´s RFID. El primero es un código de barras bidimensional binario. El segundo es por radiofrecuencia, y no involucra procesamiento de imagen.
Estas dos características hacen al proceso de identificación relativamente insensible a cambios de iluminación.

Un factor a considerar debe ser el destino de los identificadores, es decir, si la aplicación está orientada a discriminar productos para venta y distribución, el colocar los identificadores en ellos, no debe representar un gasto considerable. Por otra parte, si se trata de aplicaciones donde el identificador "volverá", se puede buscar la comodidad sin considerar gastos.

8.4 Descripción de los componentes

El proyecto consta de los siguientes componentes o subsistemas:

- Unidad RFID
- Webcam 1 Acteck ATW-650
- Computadora de escritorio
- Interfaz USB
- Robot KUKA
- Webcam 2 Logitech modelo E300

Se describirán por separado para luego explicar la interacción entre ellos y por ende, el funcionamiento del proyecto.

8.4.1 Unidad RFID

Como primera opción de identificadores serán los Tag´s RFID. Los datos serán leídos y enviados vía bluetooth a la computadora. El dispositivo encargado de ésta tarea será llamado en adelante "Unidad RFID" y se muestra en la figura 58.

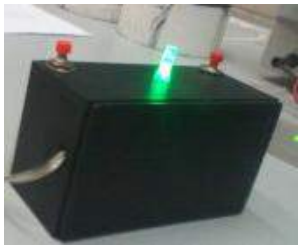

Figura 58.- Unidad RWD

La unidad RFID es la integración de un módulo lector-grabador RFID modelo **YHY502CTG**, un transmisor Bluetoothmodelo **HC-05 y** microcontrolador**PIC16F648A** en una sola unidad. Se eligió ese microcontrolador por su disponibilidad inmediata y cumplir los requerimientos de comunicación serial RS-232 e interrupción externa (detección de cambio de voltaje en un pin).

La figura 59 muestra el esquema interno de la unidad RFID. Se aprecia que existe comunicación bidireccional entre el microcontrolador PIC y el módulo de lectura-escritura (RDW). También existe

comunicación bidireccional entre el módulo RDW y los Tag´s. Y finalmente, el módulo de transmisión Bluetooth solamente transmite los resultados de las operaciones realizadas.

La descripción del funcionamiento se da a continuación:
El módulo Lector/Grabador (RWD), se encuentra constantemente emitiendo señal en busca de algún Tag. Cuando un Tag RFID entra en su rango de alcance, el RDW cambia momentáneamente el estado lógico de una de sus salidas, de ALTO a BAJO.

EL RWD, es decir, el módulo "Lector-Grabador RFID", necesita recibir los comandos correspondientes a la acción deseada. Para el alcance de éste proyecto, solamente se maneja el comando "Block Read", que leerá la información de un bloque específico del Tag. Para más información sobre la organización de la memora de un Tag, ver capítulo 4.

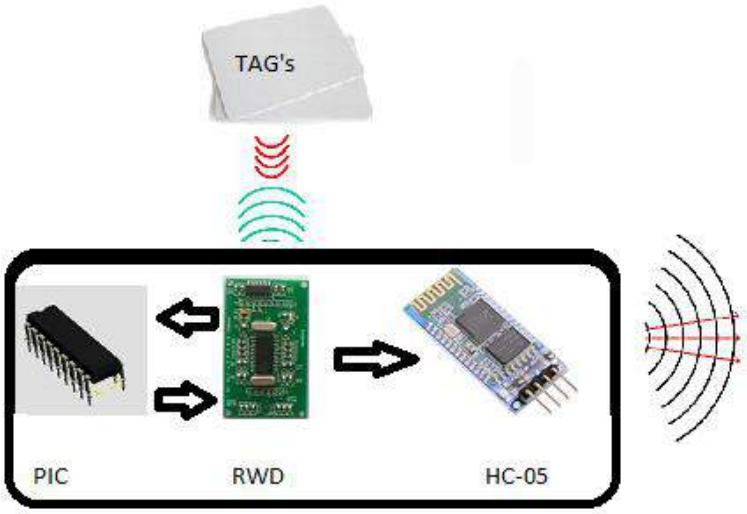

Figura 59.- Componentes de la unidad RFID

8.4.1.1 Comando "Read Block"

Send	Head	Length	Command	Key A or Key B	Block Number	Key	XOR Checksum
	AA BB	0A	21	1Byte '00': Key A '01': Key B	1Byte	6 Bytes	XOR Checksum

Figura 60.- Formato del comando "Read block"

La figura 60 muestra la descripción del comando. Todos los valores son hexadecimales. La descripción de las partes se puede ver en el capítulo 4. A continuación sólo se definirán las partes específicas para el comando "Block Read".

- "Key A or Key B" Define el tipo de Tag. Para los Tag´s utilizados debe ser (0X00).
- "Block Number" indica el número de bloque que deseamos leer. Para éste proyecto fue elegido arbitrariamente el bloque1 (0X01) Nota: El bloque 00 es un bloque de sólo lectura, por lo que no permite grabar información.
- "Key" es un password de 6 Bytes dado por el fabricante 0XFF FF FF FF FF FF.

- "XOR Checksum" Es la operación XOR entre todos los Bytes excepto "Header " y el propio "Checksum".

La operación XOR entre Bytes se hace bit a bit. Por ejemplo, sean los Bytes AB, CD y AF operandos. La figura 61 muestra el número 0XAB en binario.

	A	B
AB	1010	1011

Figura 61.- Representación binaria del Byte AB

Como referencia rápida, se muestra la tabla de verdad de la operación lógica XOR en la tabla 10

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 10.-Tabla de verdad de la operación XOR

La operación lógica XOR entre los tres operandos se encuentra haciéndola primero entre los primeros dos Bytes 0XAB y 0XCD, y luego, el resultado, se toma con el siguiente Byte (0XAF).

Es decir:

AB	⊕	10101011
CD		11001101
XOR		01100110
	⊕	
AF		10101111
XOR		11001001
		C 9

Finalmente se tiene que el resultado de la operación lógica **(0XAB)** ⊕ **(0XCD)** ⊕ **(0XAF)** es (0XC9), y se representa mediante: $AB \oplus CD \oplus AF = C9$.

Concluido el ejemplo, se tiene que los datos necesarios para calcular el Checksum de acuerdo a las especificaciónes mencionadas en la figura 55son: 0A 21 00 01 FF FF FF FF FF FF.
Para agilizar el proceso, el fabricante también incluye una aplicación de software hecha en Delphi para el cálculo del Checksum. Ésta aplicación se muestra en la figura 62.

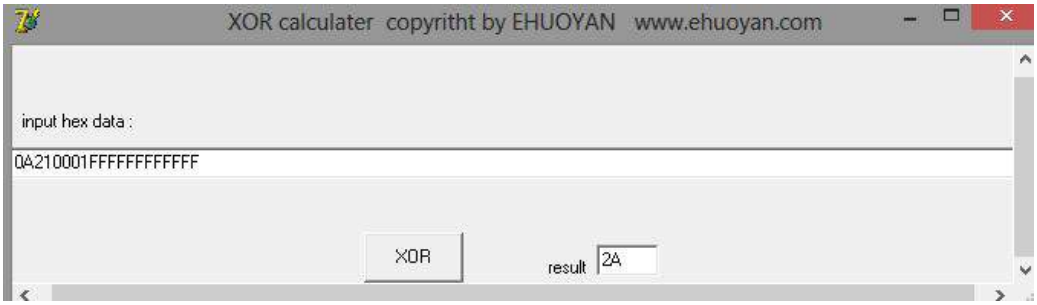

Figura 62.- Aplicación para calcular operación XOR

La ejecución de la aplicación cuya apariencia se muestra en la figura 62 muestra que el resultado es 0X2A, por tanto, el comando completo queda como: AA BB 0A 21 00 01 FF FF FF FF FF FF **2A**.

8.4.1.2Tag´s

De la gran variedad de Tag´s en el mercado, se utilizaron de tres tipos: Tarjeta, Llavero y Pulsera. Sus apariencias se muestran en la figura 50.

Los mejores resultados de lectura se obtuvieron usando las tarjetas; esto es así, debido a la mayor dimensión de la bobina de inducción. Las distancias máximas de detección correcta fueron tomadas desde el borde de la unidad lectora RFID por el método de prueba y error, y se muestran en la tabla 11.

TAG	Dist. max de lectura
Tarjeta	6cm
Llavero	5mm
Pulsera	3mm

Tabla 11.- Distancias máximas para lectura confiable

La figura 63 muestra un ejemplo de distancia de lectura.

Figura 63.- Muestra de la distancia entre la unidad de lectura y el Tag

8.4.2 Funcionamiento de la unidad RFID

En el capítulo 7 se menciona que el RDW cambia el estado lógico de uno de sus pines cuando detecta un Tag. También, que el RDW necesita recibir los comandos desde el exterior.
El propósito es, por tanto, que cuando se detecte un Tag, automáticamente se lea la información existente en el bloque 1 del mismo. Eso se consigue mediante la mediación de un microcontrolador.
El microcontrolador tiene la capacidad de responder a un cambio de estado lógico en una o varias de sus terminales. Cuando esto sucede, puede ejecutarse automáticamente una acción predefinida. A esta característica se le llama “Interrupción externa”.

Se aprovecha el cambio de estado lógico que manda el RDW, para crear una interrupción externa en el microcontrolador.

Dicho microcontrolador (PIC) se programó de tal manera, que cuando ocurra dicha interrupción, automáticamente envíe el comando "Leer tarjeta" al módulo Lector/Grabador.
El módulo RWD realizara entonces la lectura del Tag, Los datos, obtenidos de dicha lectura tienen el formato mostrado en la tabla 7, de acuerdo con el apéndice 5. Nota: la longitud de datos es hexadecimal.
Al recibir el comando, el módulo RFID ejecuta la orden y genera una respuesta vía RS-232.
En la figura 64 se muestran y explican dos resultados de la operación de lectura. Generalmente, las lecturas se ejecutan sin problemas, sin embargo hay ocasiones especiales en las que no será así. Para asegurar que no se entreguen datos erróneos al host, el módulo RDW genera un código específico para denunciar un error [34].

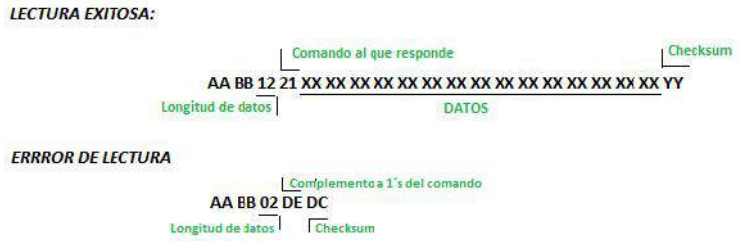

Figura 64.- Respuestas en Hexadecimal del módulo HDW.

Los errores de lectura ocurren cuando el Tag:

- No está lo suficientemente cerca.
- Se aleja del dispositivo lector durante la operación de lectura.
- Se encuentra flexionado

Estos datos obtenidos como respuesta, son enviados al módulo HC-05 para que sean transmitidos vía Bluetooth a la computadora.
El proceso se describe en la figura 65, donde se ha enumerado la secuencia del funcionamiento.

1. Cuando el Tag es detectado, se emite una señal al microcontrolador
2. Éste la toma como una interrupción y responde mandando el comando de lectura al RDW en formato RS-232.
3. El módulo RDW enciende, identifica al Tag y solicita los datos grabados en el bloque 01.
4. El Tag responde con los datos grabados en dicho bloque.
5. El RDW manda la señal RS-232 con los datos leídos al módulo HC-05.
6. Finalmente el HC-05 envía esos datos vía Bluetooth.

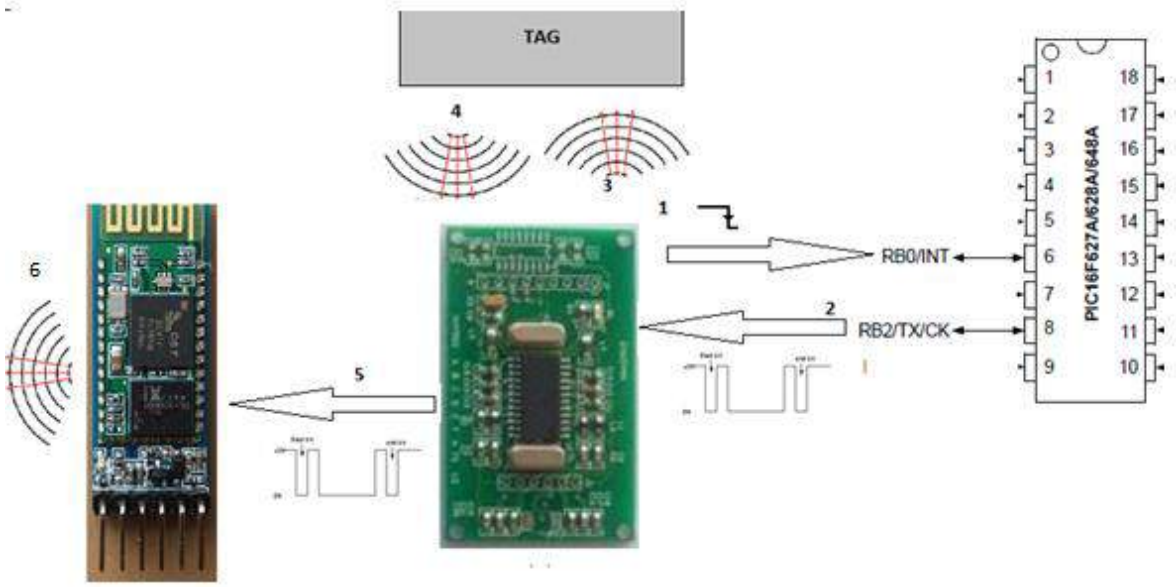

Figura 65.- Detalles del funcionamiento interno de la unidad RFID

8.5- QR

La segunda opción de identificadores es usar imágenes de código QR, y serán leídas mediante una cámara web (Webcam 2, en la figura 1). La cámara tiene el aspecto mostrado en la figura 66.

Figura 66.- Cámara web Utilizada para leer los códigos QR

Los códigos fueron generados mediante una aplicación en línea de uso libre (ver figura 67).
Dicha aplicación se encuentra en forma de una ventana en la página [37].
Y se muestra en la figura 67. En ésta aplicación, primero se escoge el tipo de dato que se quiere codificar. Puede ser:

- Dirección URL
- SMS
- Texto
- Número telefónico
- Tarjeta de identificación

Luego se procede a escribir el contenido.
Para este proyecto se utilizó el formato “Texto”, ya que se trata de datos hexadecimales.

Figura 67.- Aplicación de software para generar códigos QR

Una vez generada la imagen del código, se procedió a imprimirla, recortarla y colocarla en los objetos. La figura 68 muestra un identificador QR en tamaño real. Pertenece a un rectángulo que le corresponde el lugar marcado como "1" en la mesa de trabajo.
La información contenida en estos códigos tiene exactamente el mismo formato que la contenida en los Tag´s RFID, por tanto, el programa decodificador para las lecturas es el mismo para ambas formas de captura.
En la modalidad de códigos QR, el método de entrada de datos a la computadora es una cámara web. La estructura restante permanece sin cambios, como se aprecia en la figura 8.

Figura 68.- Ejemplo de código QR generado para un bloque rectangular

8.6 Computadora

La computadora constituye la parte central del proyecto, se encarga de la interfaz de usuario, del procesamiento de la imagen de la mesa de trabajo, de tomar decisiones en base al procesamiento de dicha imagen, de decodificar los códigos QR identificadores de objetos, de decodificar los códigos obtenidos de los Tag´s RFID, y finalmente, de dar las órdenes de movimiento al robot KUKA.
Dicha computadora tiene un sistema operativo de 32 bits, con un procesador de 3.2Ghz.
Los detalles se pueden consultar en la figura 69.

Sistema

Fabricante:	Corporativo Lanix, S.A. de C.V.
Modelo:	Lanix Computer
Evaluación:	La evaluación del sistema no está disponible
Procesador:	AMD Phenom(tm) II X4 955 Processor 3.20 GHz
Memoria instalada (RAM):	4.00 GB (3.25 GB utilizable)
Tipo de sistema:	Sistema operativo de 32 bits
Lápiz y entrada táctil:	La entrada táctil o manuscrita no está disponible para esta pantalla

Compatibilidad con Corporativo Lanix, S.A. de C.V.

Número de teléfono:	01-800-90-LANIX (01-800-90-52649)
Sitio web:	Soporte técnico en línea

Configuración de nombre, dominio y grupo de trabajo del equipo

Nombre de equipo:	CIO-PC
Nombre completo de equipo:	CIO-PC
Descripción del equipo:	
Grupo de trabajo:	WORKGROUP

Activación de Windows

Windows está activado

Id. del producto: 00359-OEM-8992687-00097

Figura 69.- Características técnicas de la PC utilizada

8.7 Interfaz USB

La computadora puede determinar los movimientos que el brazo robot debe hacer.Sin embargo, no puede manipular directamente al robot. EL robot puede interactúar con señales externas de control por medio de las entradas y salidas digitales localizadas en su consola. Para esto, la PC manda las órdenes de movimiento vía USB a un dispositivo que se encargue de decodificar la información y entregar niveles lógicos en las entradas digitales del control del robot KUKA.

Dicho dipositivo consta de un microcontrolador PIC18f4550 de Microchip, que cuenta con un módulo interno de comunicación USB y puertos de salida en formato paralelo. El dispositivo se muestra en la figura 70.

Figura 70.- Interfaz USB

El algoritmo de envío de datos por USB en la PC y el programa del microcontrolador PIC, fue desarrollado previamente por el entonces alumno Adrián Martínez González.

8.8 Robot KUKA

El brazo de robot utilizado para la manipulación de objetos es un robot industrial KUKA KR16-2 cuyas especificaciones técnicas se pueden ver en la figura 71.

KR 16-2

Gracias a su versatilidad y flexibilidad, el KR 16-2 trabaja eficazmente en la mayoría de los sectores de la industria transformadora: tanto en la industria suministradora para el sector automotriz como en otros ramos industriales.

Cargas	
Carga	16 kg
Carga adicional	10 kg
Zona de trabajo	
Máx. alcance	1610 mm
Otros datos y modelos	
Número de ejes	6
Repetibilidad	<±0,05 mm
Peso	235 kg
Posiciones de montaje	Suelo, techo
Unidades de control	KR C4

Figura 71.- Especificaciones técnicas del brazo robot [38]

Las figuras 72,73 y 74, nos muestra que los robots industriales cuentan con una unidad de control y una terminal de usuario. La unidad de control es un gabinete que contiene todos los circuitos de control para el robot, mientras que la terminal de usuario se encarga de la gestión de los programas que ejecutará el robot; pero principalmente, es la terminal de programación.

La unidad de control KUKA (Figura 68), cuenta con entradas y salidas digitales, que sirven para interactuar con sistemas externos.

Figura 72.- Brazo robot KUKA

Figura 73.- Gabinete o consola de control del brazo robot

Figura 74.- Terminal de usuario

En el laboratorio de robótica, se construyó previamente un gabinete adicional para la gestión de dichos puertos o entradas digitales (figura 75). Es ahí a donde llegan las órdenes del micro controlador de conexión USB en formato paralelo. El gabinete tiene la peculiaridad de poder elegir la fuente de excitación para cada una de sus entradas digitales. Para ésta aplicación fueron útiles las opciones “PIC” y “Manual”.
La opción “Manual”, fue de gran utilidad para este proyecto, ya que, por medio de ella es posible manipular el valor de las entradas digitales para hacer pruebas con el programa que se está trabajando, sin la necesidad de encender la computadora ni conectar la interfaz USB (Figura 75).

Figura 75.- Gabinete de entradas y salidas del robot

Para el presente proyecto se utilizaron 4 entradas digitales de 12 disponibles, lo que arroja un margen de 16 órdenes para el robot (2^4).
El programa del robot consiste principalmente en movimientos “punto a punto”, y control de la salida digital #3, que controla pinza del robot.
La siguiente tabla (tabla 12), muestra los códigos digitales y los movimientos programados por el autor para ellos.

Numero Decimal	Binario	Movimiento
0	'"000000000000"	Posición de descanso
1	'"000000001000"	Colocar Pieza 1 en la mesa
2	"000000000100"	Colocar Pieza 2 en la mesa

3	"000000001100"	Colocar Pieza 3 en la mesa
4	"000000000010"	Colocar Pieza 4 en la mesa
5	"000000001010"	Colocar Pieza 5 en la mesa
6	"000000000110"	Colocar Pieza 6 en la mesa
7	"000000000110"	Colocar Pieza 7 en la mesa
8	"000000000001"	Colocar Pieza 8 en la mesa
9	"000000001001"	Colocar Pieza 9 en la mesa
10	"000000000101"	Colocar Pieza 10 en la mesa
11	"000000001101"	Colocar Pieza 11 en la mesa
12	"000000000011"	Colocar Pieza 12 en la mesa
13	"000000001011"	Coger Pieza
14	"000000000111"	Colocar Piezas para ajustar la cámara
15	"000000001111"	Retirar Piezas para ajustar la cámara

Tabla 12.- Funciones programadas en el robot para las entradas digitales

8.9 CODIFICACION DE LA INFORMACION EN LOS TAG´S

Los datos leídos en cada identificador constan de 16 Bytes. Para el alcance del presente proyecto se utilizan los primeros y los últimos. Por ejemplo: **01FFFFFFFFFFFFFFFFFFFFFFFFFF04.**
El primer Byte corresponde al tipo de pieza. El último indica el lugar que ocupara la pieza en la mesa de trabajo.
En la tabla 13 se muestra la decodificación del primer Byte.
La figura 76 muestra la interpretación del último Byte. Se representa la superficie de la mesa de trabajo dividida imaginariamente en doce espacios, mismos que serán ocupados por los objetos. Cabe mencionar que los lugares 1, 2, 3 y4, en la misma figura 76, están del lado del robot. Ver también figura 1.
La razón para haber dejado sin información relevante la parte central del número, radica en la libertad de asignar ahí algún uso futuro sin modificar demasiado el programa que lo interpreta.

NUMERO	PIEZA
01	PARALELEPIPEDO
02	CUBO
03	CILINDRO
04	PRISMA TRIANGULAR

Tabla 13.- Codificación del tipo de Objeto

Figura 76.- Asignación de posiciones en la mesa de trabajo

La finalidad de la información contenida tanto en los códigos QR como en los Tag´s, es que los objetos puedan acomodarse como se ilustra en la siguiente figura (Figura 77), donde se puede apreciar la información fusionada de las dos tablas anteriores (tabla 13 y figura 76).

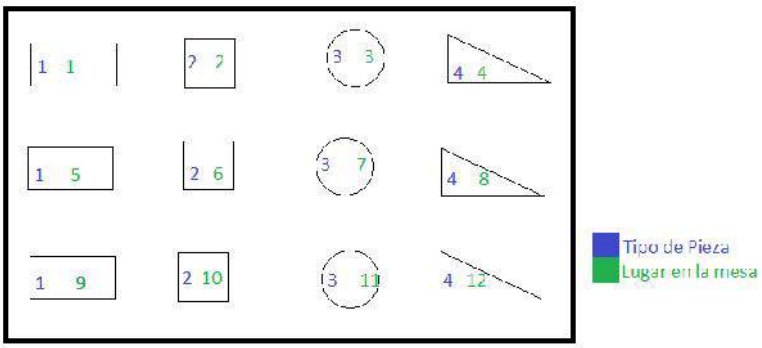

Figura 77.- Objetos ordenados en la mesa de trabajo. Incluyen la información grabada en sus identificadores. (El primer número indica el tipo de pieza –columna- y el segundo indica el lugar en la mesa de trabajo)

En la figura 77 se puede observar que los bloques cuyo primer número es "1" son rectángulos, y se colocan en la primer columna. Los bloques cuyo primer número es "2" son cuadrados, y se colocan en la segunda columna, y así sucesivamente.
También se deriva de la figura 77, que los bloques de una columna no están dispuestos al azar. El segundo número de cada bloque indica su lugar en la mesa de trabajo. (ver las distribuciones de la figura 76).

8.10PROCESO

8.10.1Descripción del procedimiento

Se supone una banda transportadora que pondrá a los bloques en posición de ser detectados. Como ya se ha mencionado con aterioridad, los identificadores de los objetos serán leídos por el modo elegido por el usuario. Esa información llegará a la computadora para ser procesada. La computadora hace ése trabajo por medio del software MATLAB.

La funcion del programa hecho en Matlab, es recibir la lectura del objeto en turno, decidir que lugar ocupara en la mesa de trabajo, y dar la orden al robot para ejecutar los movimientos de piezas.

Se controlan dos webcams, una para tomar imágenes de la mesa de trabajo, la otra para leer lo codigos QR en caso de ser requerido.

El programa, al recibir una lectura valida -(via QR o RFID)- , toma una imagen de la mesa de trabajo.
Se aplica un procesamiento de imagen llamado rectificación, para determinar con mayor presición la posicion de los objetos que se encuentran sobre la mesa de trabajo.

Existen dos modalidades para el acomodo de piezas: (ver figuras 78 y 79)

- Colocar las piezas en el lugar designado en su identificacion, previa confirmacion de que el lugar se encuentra vacio.
- Acomodar las piezas; ocupando en orden, los lugares vacios en la mesa .

En la primera modalidad, el sistema respeta el lugar asignado a cada objeto (o pieza),que viene grabado en su identificador.
Si una pieza trae grabado un lugar que, en ese momento se encuentre ocupado, el robot simplemente no tomara la pieza; y aparecerá la interfaz de usuario, la leyenda “No hay Lugar disponible”. En la figura 78, se muestran los dos casos. En el primer caso, la pieza tiene asignado el lugar “1”. Como dicha posicion se encuentra vacante, colocará ahí la pieza. En el segundo caso,la pieza viene con la posición #6, que, esta ocupado. Como ya se mencionó, el robot no tomará acción alguna.

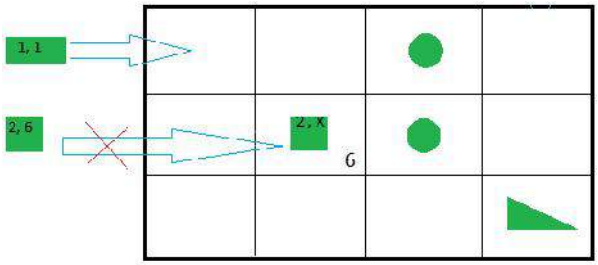

Figura 78.- EL primer número en el identificador indica el tipo de pieza, el segundo indica el lugar en la mesa de trabajo. La figura indica el lugar que deberían ocupar los objetos en la mesa de trabajo. El lugar del rectángulo está vacante, mientras el lugar del cubo no lo está.

.
En cambio,en la segunda modalidad, solamente tomara en cuenta que tipo de pieza es, y la acomodará en la columna correspondiente, ocupando algún lugar que en ese momento se encuentre vacio.

En caso de que se encuentren vacíos dos lugares, e incluso los tres; el programa elegirá el lugar mas cercano a la parte superior de la mesa. Esto se puede ver en la figura 79.

En dicha figura, se observa la columna de rectángulos vacía; por lo que al recoger un rectangulo, el sistema lo asigna a la parte superior de la mesa..
Si el objeto es un cubo, el procesamiento previo de la imagen arrojara que sólo existe disponible el espacio intermedio de la columna, por lo que allí se mandará colocar la pieza.
Por otro lado, si la pieza es un triángulo, el procesamiento arrojará dos lugares disponibles. Por consiguiente, el programa elegirá el lugar mas cercano a la parte superior de la mesa para colocar la pieza.

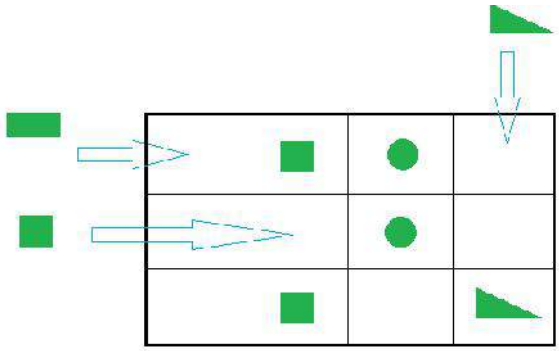

Figura 79.- En la segunda modalidad, sólo importa el tipo de objeto. La pieza se acomoda en su columna correspondiente, dando preferencia a la parte superior de la mesa.

Adicionalmente, durante la ejecución de movimientos por parte del robot, deben administrarse correctamente los intervalos de las órdenes enviadas. Ya que si se mandan varias órdenes en poco tiempo, el robot no las ejecutará todas.
Es necesario conceder tiempos de espera entre las órdenes dadas al robot, tomando en cuenta los tiempos de ejecución de éste, para que no se pierdan movimientos.
La pérdida de movimientos por parte del robot se explica a continuación:

Sea la secuencia de órdenes para el robot: "movimiento1 – movimiento2 – movimiento3".
Si se perdieran movimientos, probablemente sólo se ejecutará: "Movimiento1 – Movimiento 3".
Es decir, se ignorarán movimientos intermedios.

Esto es, debido a que el robot no tiene memoria de las órdenes que han sido enviadas a sus entradas digitales. Al terminar de ejecutar una orden, simplemente ejecutará la orden que se encuentra presente en ese momento en dichas entradas.
La figura 80 ilustra en forma gráfica lo anterior. Se representan bajo el nombre de "Dato X" los intervalos de tiempo en los que se encuentra presente una combinación lógica digital en las entradas del gabinete de control.
Igualmente, se representan bajo el nombre de "Ejecución X" los intervalos de tiempo que toma al robot ejecutar un movimiento como respuesta a las entradas digitales. Para un correcto funcionamiento, hay que ser cuidadosos de que cada que se termine una ejecución por parte del robot, los valores digitales en las entradas del mismo, sean las del movimiento siguiente. Se proponen tiempos hipotéticos para ilustrar el proceso.

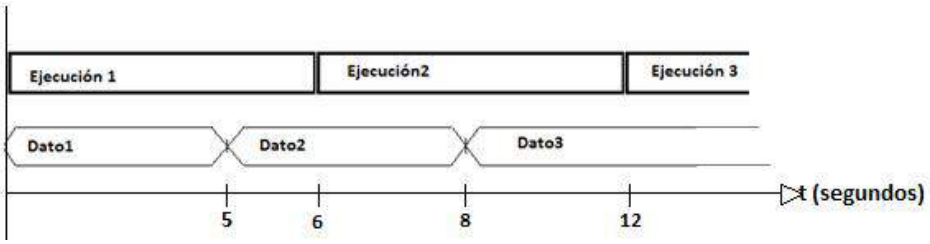

Figura 80.- Sincronización entre las entradas digitales del robot y los tiempos de ejecución de movimientos

De la figura 80, se observa que el robot es capaz de terminar la primera orden recibida en 6 segundos. Sin embargo, el número binario en las entradas digitales cambió a los 5 segundos. Esto no representa problema alguno. Después de los 6 segundos, el robot ejecuta el segundo movimiento, que le toma otros 6 segundos. Una vez iniciada una ejecución, no importa que el estado lógico de las entradas digitales cambie. EL robot terminará de ejecutar el movimiento.
Sin embargo, si se mandara una tercera orden a sus entradas antes de que termine de ejecutar el primer movimiento, el robot ya no ejecutara la segunda orden, sino la tercera.
Esto se muestra en la figura 81, donde, al aumentar la velocidad de envío de datos, se pierde el movimiento 2, ya que al terminar la "Ejecución1" a los 6 segundos, el dato presente en las entradas es el "Dato3", y no el "Dato 2"

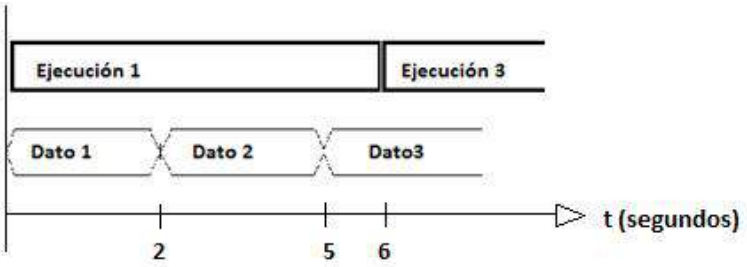

Figura 81.- Desincronización entre los tiempos de ejecución del robot y sus entradas digitales

En la figura 82 se muestra el diagrama de bloques del programa descrito hasta el momento.

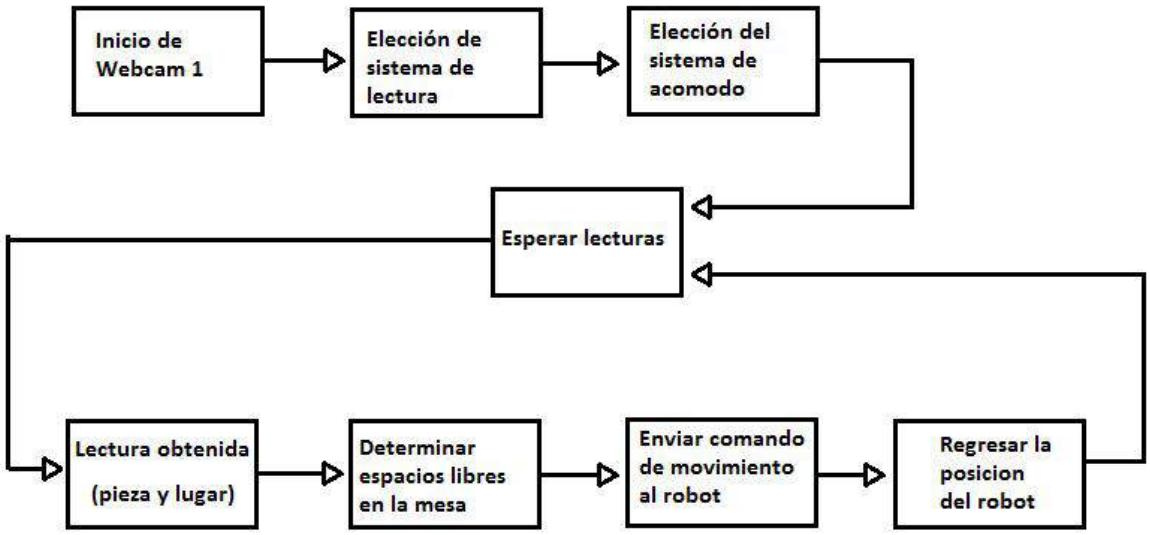

Figura 82.- Diagrama de flujo del programa en MATLAB

8.11 MATLAB

El programa desarrollado está hecho con la herramienta GUIDE y la versión R2012b de MATLAB. La aplicación hecha en GUIDE, se muestra en la figura 83 antes de correr el programa.

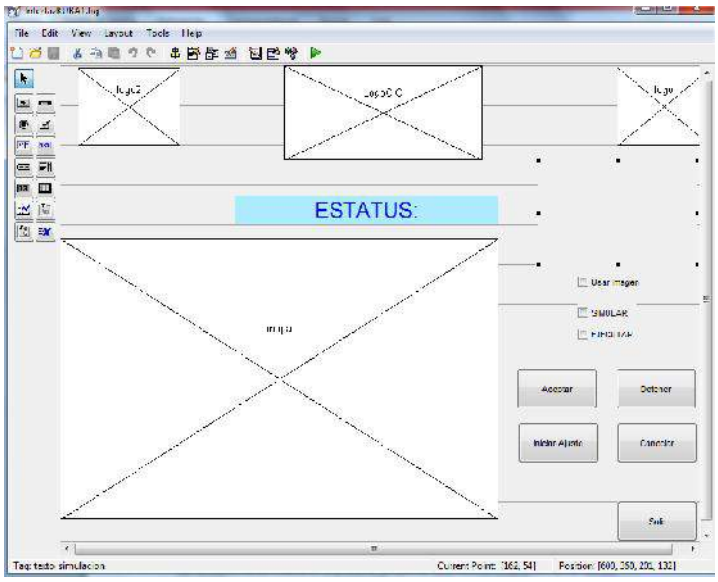

Figura 83.-interfaz de usuario en GUIDE

La programación requirió el desarrollo de algunas funciones, con la finalidad de hacer más amigable la programación; así como de identificar más fácilmente algún problema.

Algunas funciones ya existían en la computadora, como, por ejemplo, "PIC_KUKA(….)", que fue desarrollada previamente, y como las funciones para identificar y decodificar imágenes QR, que se encuentra en la red como software libre.

Las funciones desarrolladas para el presente proyecto se enumeran a continuación como parte del funcionamiento del mismo, y hacen uso de esas útiles herramientas.

Al iniciar el programa, se hacen invisibles la mayoría delos objetos del GUIDE. Se harán visibles, de acuerdo a las opciones que el usuario vaya eligiendo.
Cuando el usuario elige un modo de detección, se carga la cámara de la mesa de trabajo.

8.11.1RECTIFICACIÓN DE IMAGEN

Contando con el modelo de transformación descrito en el capítulo 6, es necesario determinar los puntos que se tomarán como referencia para el cálculo de las matrices de distorsión.

Se muestra en la figura 84, una imagen obtenida por la webcam1, que se utilizó en el proyecto. El patrón cuadriculado se dibujó sobre una hoja de papel Bond y mide 59cm X 88cm.

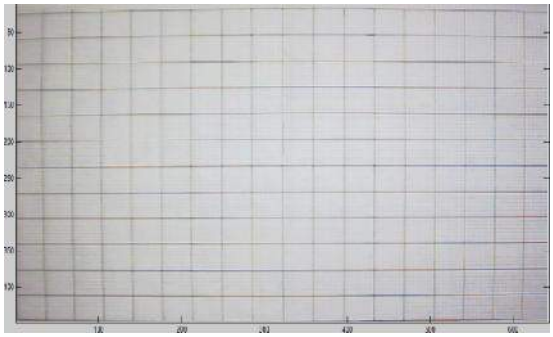

Figura 84.- Imagen original

Dichos puntos de referencia se tomaron en los cruces de las líneas más obscuras como se muestra en la figura 85. La finalidad de la calibración de la cámara en éste proyecto no incluye aspectos de metrología óptica, por lo que no interesan los puntos P_w de un punto en el mundo real, tamaño de pixel o distancia focal. Sólo interesan los puntos (X_c, Y_c) y (X_u, Y_u), que son los puntos real e ideal, respectivamente en el plano de imagen. La definición formal de estos parámetros se encuentra en el capítulo 6.

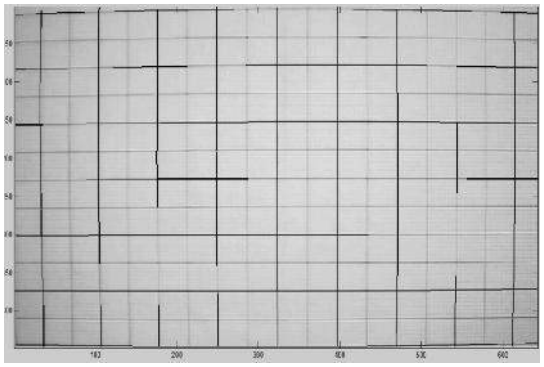

Figura 85.- Imagen original con remarcaciones

El tipo de deformación que se quiere corregir se acentúa en los bordes de la imagen, por lo que el cuadro central es el más fiable y se tomó como referencia para trazar una cuadrícula ideal sin deformación en MATLAB a partir de él. Esto se muestra en la figura 86

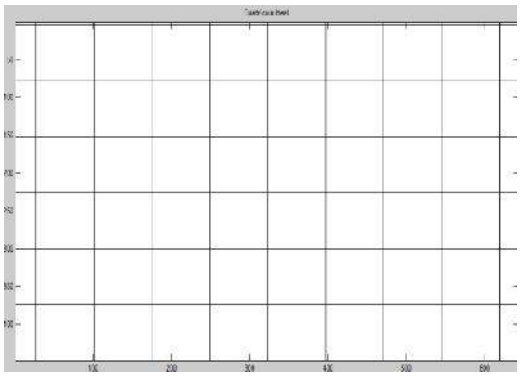

Figura 86.- Patrón cuadricular ideal a partir de un cuadro en el centro de la imagen original

Para obtener una imagen conveniente a partir de la figura 85, primero se hace una binarización de la imagen. Dicho proceso consiste en asignar valores binarios a cada pixel de la imagen, respecto a un nivel conocido como umbral. Es decir:

$$im(x,y) = \begin{cases} 0 & si\ f(x,y) > T \\ 1 & si\ f(x,y) \leq T \end{cases} \qquad \text{ec. 28}$$

Donde T es el nivel de umbral, f(x,y) es la intensidad del punto (x,y) y p(x,y) es alguna propiedad local del punto.
Cuando T depende sólo de f(x,y), el umbral se llama *global*. Si T depende tanto de f(x,y) como de p(x,y) entonces el umbral se llama *local*. Si T depende de las coordenadas espaciales x e y, se llama *umbral dinámico* [31].
Para el procesamiento de la imagen en el presente proyecto se utilizó un umbral total debido a que solamente es necesario detectar la presencia de objetos y a la uniformidad en la iluminación.

Por lo que se utilizó el siguiente código de MATLAB mostrado en la figura 87.

```
image=imread('cuadriculaUSB2_5.jpg', 'jpg'); %Cargar imagen
[y,x]=size(image);                            %Tomar las dimensiones de la imagen
bw = im2bw(image,0.3);                        %Binarizar la imagen

%%
BW2=~bw;                          %invertir los niveles lógicos de la imagen
BW3 = bwmorph(BW2,'skel',Inf);    % filtro de esqueleto
```

Figura 87.- Código en MATLAB para binarizar la imagen y aislar las líneas marcadas

El resultado es la imagen mostrada en la figura 88.

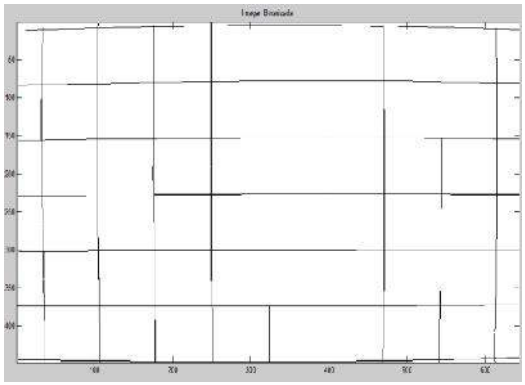

Figura 88.- Resultado de binarizar la imagen original

A continuación, es necesario buscar los puntos de cruces de líneas en las dos imágenes (ver figura 86 y figura 88) y representarlos en un mismo plano. Dichos puntos de cruce darán el comportamiento de la deformación a través del área de la imagen.

La forma de encontrar los puntos de cruce es la siguiente: Se hace un recorrido vertical descendente buscando un "1" lógico.
Cada que sea encontrado alguno, se pregunta si existe otro "1" lógico colocado a la derecha.
Si es así, se comienza a hacer una "búsqueda de vecinos". Si se encuentran más de tres, entonces se trata de un cruce de líneas. En total se encontraron 63 puntos de cruce.

En la figura 89 se ilustra éste procedimiento. Los cuadros representan los pixeles examinados de una columna. Los pixeles marcados con un circulo vacío simbolizan "candidatos" a cruce de línea. El pixel marcado con un circulo relleno muestra un "candidato" que sí resultó ser un cruce de línea.

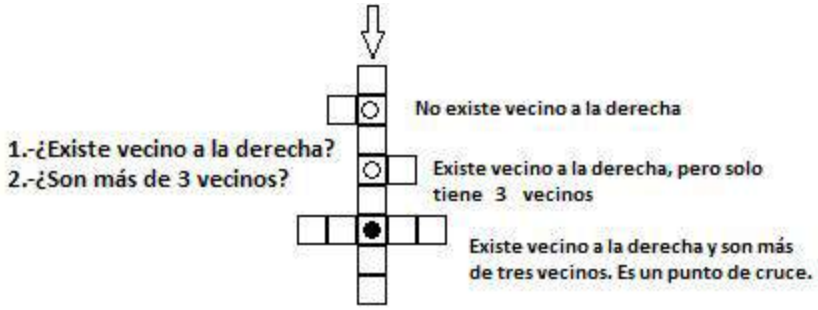

Figura 89.- Algoritmo de búsqueda de cruces de línea

Puede suceder que un punto de cruce de líneas no esté perfectamente alineado. Sin embargo, debido a que los vecinos los busca en todas direcciones, el algoritmo sigue funcionando.
En la figura 90 se ejemplifica lo anterior. Un punto pequeño representa los lugares en donde se buscan los vecinos.
Por tanto, se puede apreciar que el pixel marcado con un circulo relleno, también se reconoce como punto de cruce por habérsele encontrado más de tres vecinos.

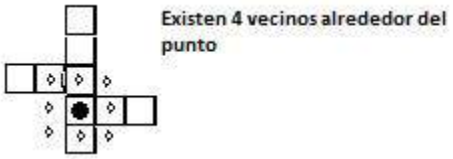

Figura 90.- Cruce de líneas atípico identificado

Se muestra en la figura 91 el algoritmo de MATLAB para determinar los puntos de cruce de las líneas de la imagen real. La imagen real ya binarizada tiene el nombre de "BW3"

```
%Generación de puntos de cruce
imagenpuntos=zeros(y,x);
contador =0;
for i=2:x-2
    for j=2:y-1
        if ((BW3(j,i)==1)&&(BW3(j,i+1)==1))
            for a=1:-1:-1
                for b=1:-1:-1
                    if(BW3(j+a,i+b)==1)
                        contador=contador+1;
%                           imagenpuntos(j,i)=1;
                    end
                end
            end
            if (contador>3)
                imagenpuntos(j,i)=1;
            end

        end
        contador=0;
    end
end
```

Figura 91.- Código para buscar puntos de cruce

El resultado del algoritmo se muestra en la figura 92. Es evidente que los puntos centrales coinciden a la perfección, mientras que los puntos cercanos a los bordes tienden a separarse.

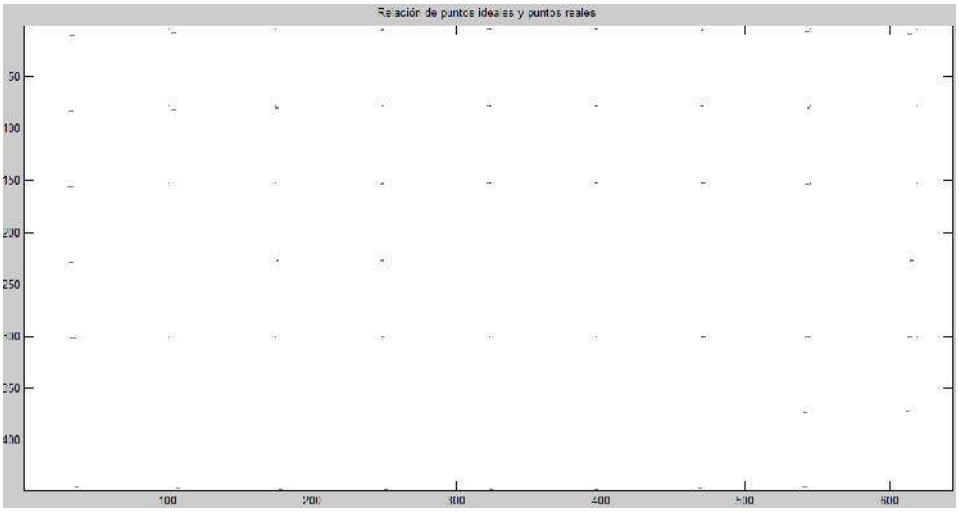

Figura 92.- Puntos de cruce real y de cruce ideal

La figura 93 es un acercamiento a la esquina superior derecha de la figura 92. Se puede apreciar con mayor claridad la discrepancia entre los puntos simples y los puntos representados como “Barras” pequeñas. Los puntos simples son los cruces de líneas de la imagen ideal, mientras que las “Barras” representan los cruces de las líneas reales.

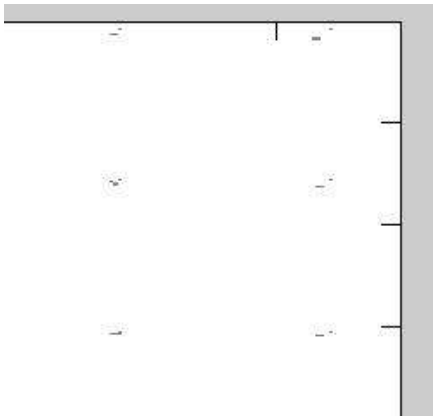

Figura 93.- acercamiento a un borde de la imagen

La diferencia de coordenadas entre los puntos reales e ideales, darán la deformación de la imagen original. Dichas diferencias arrojarán una matriz para las diferencias en "X", y otra matriz para las diferencias en "Y".
El ordenamiento de los puntos sigue la lógica mostrada en la figura 94.

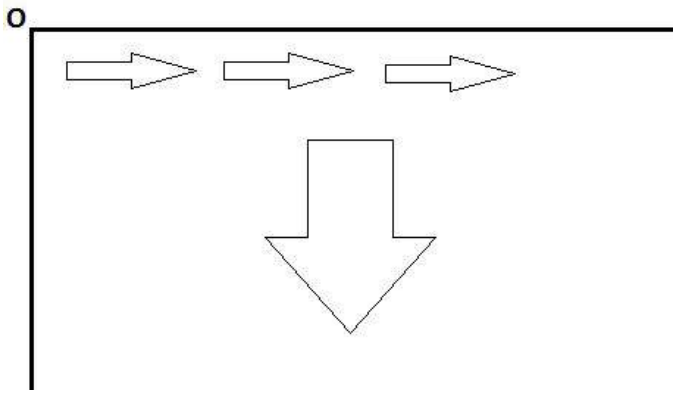

Figura 94.- Orden de búsqueda de puntos de cruce

Comenzando desde el origen de la imagen (Figura 94), se hace un recorrido horizontal hasta el final de la imagen, para luego repetir la operación en el renglón inferior y así, sucesivamente.

A continuación, se muestra la lista de los puntos de cruce de línea de las dos imágenes:
Comparten el mismo renglón los puntos de cruce que deberían coincidir. Por ejemplo, según el acercamiento mostrado en la figura 95.

El primer punto ideal está localizado, en la posición (x=27, y=4).
El segundo punto ideal está localizado, en la posición (x=101, y=4).

De acuerdo a la misma figura:

El primer punto real está localizado en la posición (x=34, y=10).

El segundo punto real está localizado en la posición (x=104, y=8).

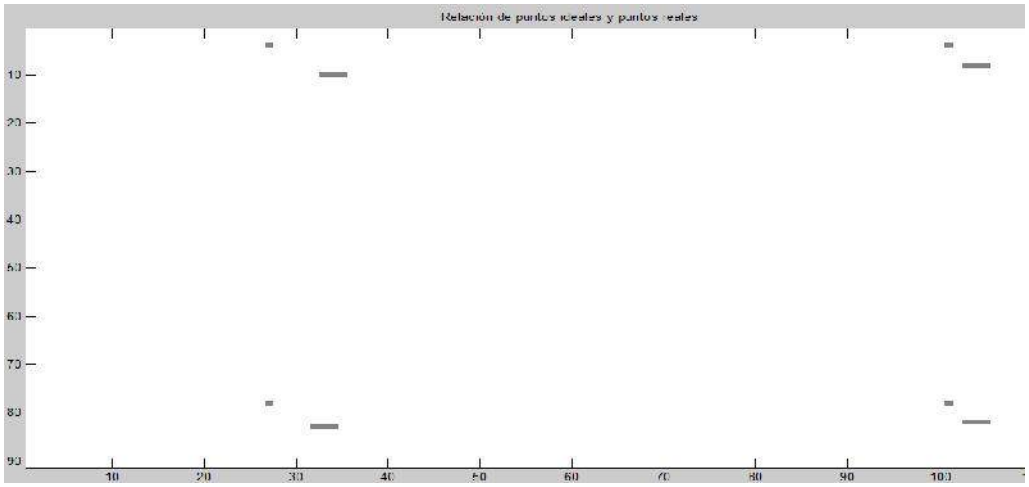

Figura 95.- Identificación de los primeros dos pares de puntos de cruce.

Nótese que la distancia entre el primer punto real y el primer punto ideal es, de acuerdo al teorema de Pitágoras:

$$d = \sqrt{((\Delta x)^2 + (\Delta y)^2)} \qquad \text{ec. 29}$$

Por tanto: $d = \sqrt{((-7)^2 + (-6)^2)} = 9.21954446$, como se puede observar en la tabla 14

El cálculo de ésta distancia fue de gran utilidad para determinar la correspondencia entre los puntos de cruce encontrados. Ya que debido a la relativamente poca deformación, los puntos correspondientes son los que se encuentran más cerca entre sí.

Punto de Cruce #	IMAGEN IDEAL		IMAGEN REAL		PARÁMETROS		
	X	Y	X	Y	DISTANCIA	DELTAX	DELTAY
1	27	4	34	10	9.21954446	-7	-6
2	101	4	104	8	5	-3	-4
3	175	4	176	6	2.23606798	-1	-2
4	249	4	249	5	1	0	-1
5	323	4	323	4	0	0	0
6	397	4	397	4	0	0	0
7	471	4	470	5.33333	1.66666667	1	-1.333
8	545	4	543	7	3.60555128	2	-3
9	619	4	614	9	7.07106781	5	-5
10	27	78	33	83	7.81024968	-6	-5
11	101	78	104	82	5	-3	-4
12	175	78	176	80	2.23606798	-1	-2

13	249	78	249	79	1	0	-1
14	323	78	323	78	0	0	0
15	397	78	397	78	0	0	0
16	471	78	470	78	1	1	0
17	545	78	543	79.6667	2.60341656	2	-1.667
18	619	78	615	81	5	4	-3
19	27	152	32.5	156	6.80073525	-5.5	-4
20	101	152	104	154	3.60555128	-3	-2
21	175	152	176	154	2.23606798	-1	-2
22	249	152	249	153	1	0	-1
23	323	152	323	152	0	0	0
24	397	152	397	152	0	0	0
25	471	152	471	152	0	0	0
26	545	152	543.5	153	1.80277564	1.5	-1
27	619	152	615	154	4.47213595	4	-2
28	27	226	33	229	6.70820393	-6	-3
29	101	226	104	228	3.60555128	-3	-2
30	175	226	176	227.333	1.66666667	-1	-1.333
31	249	226	249	227	1	0	-1
32	323	226	323	226	0	0	0
33	397	226	397	226	0	0	0
34	471	226	471	226	0	0	0
35	545	226	544	226	1	1	0
36	619	226	615	227	4.12310563	4	-1
37	27	300	34.5	302	7.76208735	-7.5	-2
38	101	300	105	301	4.12310563	-4	-1
39	175	300	177	301	2.23606798	-2	-1
40	249	300	249	301	1	0	-1
41	323	300	323	301	1	0	-1
42	397	300	397	301	1	0	-1
43	471	300	471	300	0	0	0
44	545	300	543	300	2	2	0
45	619	300	614	300	5	5	0
46	27	374	36	374	9	-9	0
47	101	374	106	374	5	-5	0
48	175	374	177	374	2	-2	0
49	249	374	250	374	1	-1	0
50	323	374	323.5	374	0.5	-0.5	0
51	397	374	397	374	0	0	0
52	471	374	470	374	1	1	0
53	545	374	542	373	3.16227766	3	1
54	619	374	613	372	6.32455532	6	2
55	27	448	37	445	10.4403065	-10	3
56	101	448	107	446	6.32455532	-6	2
57	175	448	178	447	3.16227766	-3	1
58	249	448	251	447	2.23606798	-2	1

59	323	448	324	447	1.41421356	-1	1
60	397	448	397	447	1	0	1
61	471	448	469	446	2.82842712	2	2
62	545	448	541	445	5	4	3
63	619	448	612	443	8.60232527	7	5

Tabla 14.- Lista de puntos de cruce y su relación con los puntos ideales

La tabla 14 muestra los puntos ideales relacionados con los puntos reales correspondientes. Se incluye la distancia entre ambos (ideal y real) en pixeles, así como la diferencia en el eje "X" y la diferencia en el eje "Y".

De la misma tabla se extraen las diferencias en "X" y se acomodan en una matriz. Matlab, mediante el comando "Surf" puede interpretar los datos de una matriz como niveles de una superficie. Dicha superficie se muestra en la figura 96

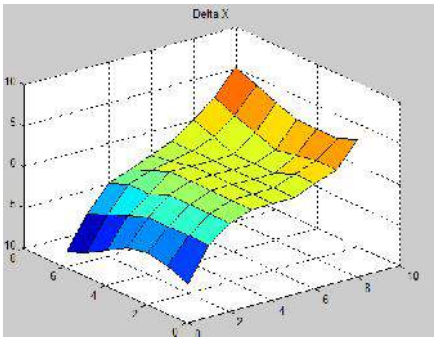

Figura 96.- Deformación de la posición X

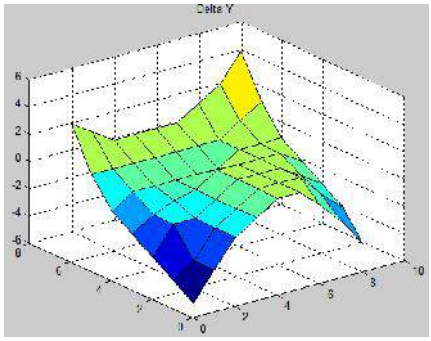

Figura 97.- Deformación de la posición y

Se procede de la misma forma para obtener la imagen de la figura 92, que corresponde a las diferencias en "Y".

Al re escalar estas matrices al tamaño original de la imagen, se obtienen los factores de corrección de posición para cada pixel.
A continuación, se muestran las imágenes original y corregida al hacer el procedimiento descrito.

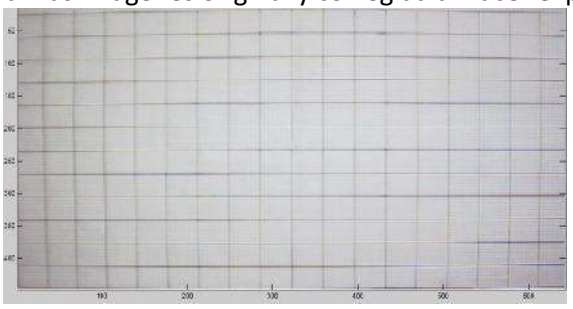

Figura 98.-Imagen a procesar

En la imagen original (figura 98), se aprecia la deformación debida a la lente de la cámara (tipo barril), al contar con la matriz de corrección, se tiene una referencia para el reacomodo de los pixeles.

Ahora, en la imagen corregida, en la posición (j,i) no se colocará la información correspondiente al pixel (j,i) de la imagen original. Si no la información del pixel (j-delta*y* , i-delta*x*) de dicha imagen original.

En la figura 99 se muestra el código en MATLAB para el reacomodo de pixeles.

```
%% Corrección
delta3y=imresize(deltay,[448,643]);
delta3x=imresize(deltax,[448,643]);

for i=1:603
    for j=1:438
        corregida(j,i)=image(round(j-delta3y(j,i)),round(i-delta3x(j,i)) );
    end
end
figure;
imagesc(image);
title('Imagen Original');
colormap gray;
 figure;
 imagesc(corregida);
 title('Corregida');
 colormap gray
```

Figura 99.- Código de corrección de imagen

Aplicado el algoritmo, la imagen resultante se puede apreciar en la figura 100.

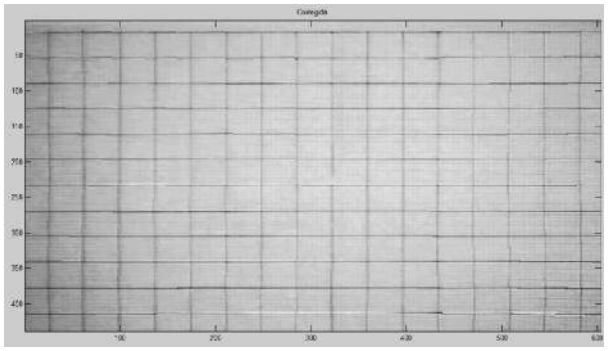

Figura 100.- Imagen corregida

El proceso descrito de calibración de la cámara no se repite cada vez que se captura una imagen con la webcam1.

Como se trata de la misma cámara en la misma posición, se guarda la matriz de corrección en un archivo. Cada vez que haya que rectificar una nueva imagen, simplemente se llama a la matriz de corrección para hacer el proceso de reacomodo de pixeles.

El procesamiento de imagen se hace sólo en 2D, ya que la cámara permanece fija a una distancia constante de la mesa de trabajo. Esto hace irrelevantes los cálculos de profundidad y minimiza la necesidad de correcciones proyectivas; aspectos propios del análisis en 3D. Como es evidente en

las figuras 101 y 102, la parte superior de los bloques está pintada de color negro. Esto fue realizado previamente a éste trabajo de tesis, y tiene la finalidad de resaltar solamente la superficie superior de los bloques para facilitar su análisis en 2D.

Al capturar una nueva imagen, se capta en formato RGB (Figura 101). El primer paso es transformarla a escala de grises (Figura 102).

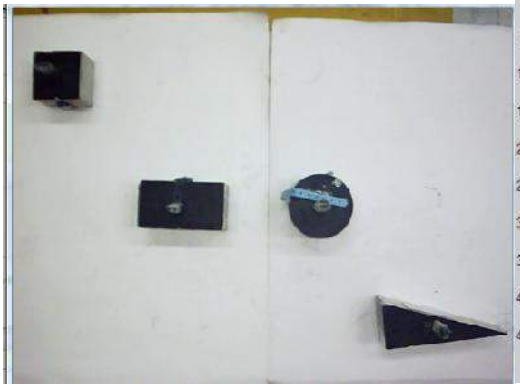

Figura 101.- Imagen original a color

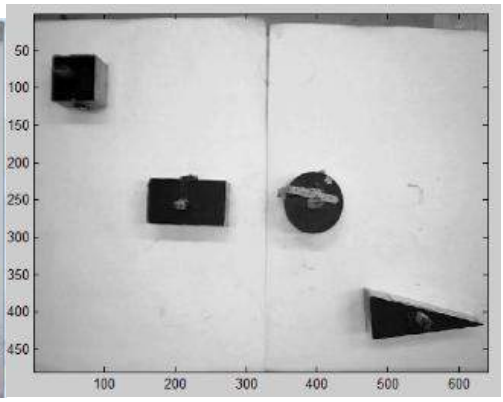

Figura 102.- Imagen en escala de grises

Luego se hace directamente la corrección de rectificación. Hasta éste punto la imagen tiene el aspecto mostrado en la figura 103.

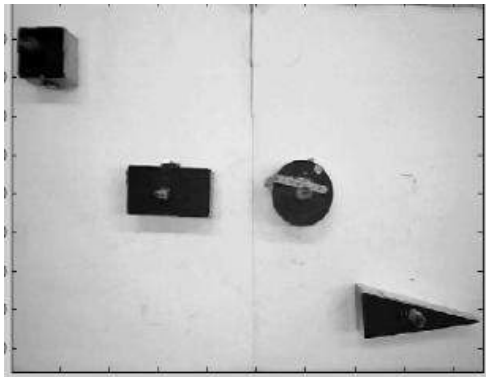

Figura 103- Imagen rectificada

8.11.2 Detección de Objetos

El siguiente paso es binarizar la imagen corregida; es decir, tomar un nivel de gris como referencia, y utilizarlo como un umbral.

Los valores de gris por debajo de ese umbral, tomarán el valor lógico de “0”.
Los valores de gris por encima de ese umbral, tomarán el valor lógico de “1”.

Como resultado se tiene una imagen de sólo dos valores en sus pixeles, y se le conoce como Imagen binaria. El resultado de éste proceso se muestra en la figura 104.

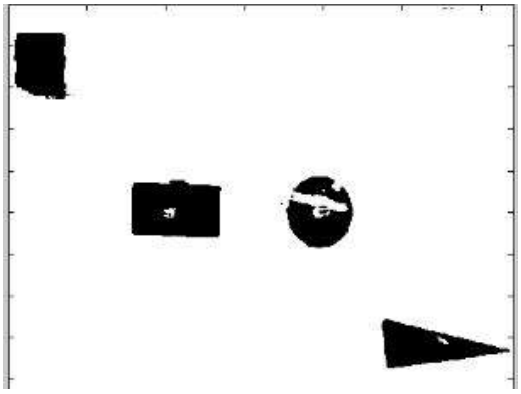

Figura 104.- Imagen binarizada

Luego se hace una inversión de la imagen. Es decir, los pixeles son sometidos a una operación lógica "NOT". El resultado es una imagen donde lo que antes era negro, ahora es blanco y viceversa. Ver figura 106.

En el siguiente paso se rellenan huecos y se eliminan áreas menores a 100 pixeles. Después del dicho procedimiento, quedan algunas áreas (en blanco) que corresponden a cada objeto.
Hasta éste punto, la imagen tiene la apariencia mostrada en la figura 106.

La finalidad de haber invertido la imagen es que las "áreas" que reconoce MATLAB son espacios con valor lógico ALTO ("1"´s).

La ventaja de que MATLAB reconozca áreas, es que se pueden aprovechar las herramientas de la función "regionprops" como "Centroids", que arroja una lista de los centroides de las áreas sin mayor dificultad.

El centroide $(\bar{x},\bar{y})$ es una propiedad métrica de una figura, y es un único punto representativo de la misma.
En el procesamiento de imagines y en visión por computadora, los momentos centrales y estadísticos son contienen información importante de una imagen.
Los momentos estadísticos en forma discreta se definen de acuerdo a [39] como:

$$m_{pq} - \sum_{i=0}^{u-1}\sum_{j=0}^{v-1} x_i^p y_j^q f(x,y),$$

ec. 30

Donde *f(x,y)* es la intensidad del pixel y *(x,y)* son las coordenadas del pixel. Los momentos centrales se definen como

$$\mu_{pq} = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} (x_i - \bar{x})^p (y - \bar{y})^q f(x,y),$$

ec.31

Donde los momentos de orden uno, m_{01} y m_{10} junto con m_{00} determinan el llamado centro de gravedad del objeto; por tanto:

$$\bar{x} = \frac{m_{10}}{m_{00}} \qquad \bar{y} = \frac{m_{01}}{m_{00}}.$$

ec. 32

Son las coordenadas del centroide de una figura.

Adicionalmente, los momentos normalizados se definen como:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}, \quad \gamma = \frac{p+q}{2} + 1.$$

ec.33

Las líneas de código correspondientes se muestran en la figura 105.

```
%% Centroides
   s  = regionprops(bw, 'centroid');
   centroids = cat(1, s.Centroid);
```

Figura 105.- Código para encontrar centroides

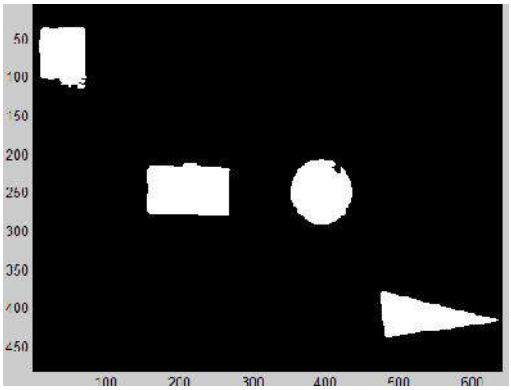

Figura 106.- Inversión y limpia de la imagen binarizada

Luego se marcan las divisiones en la imagen. Debido a que la cámara tendrá la misma orientación, éstas se pueden considerar constantes
También se procede a remarcar los centroides de las áreas encontradas por el programa con una cruz de longitud de 5 pixeles en cada dirección. El código es el mostrado en la figura 107

Dependiendo del nivel de binarización aplicado puede parecer que algún objeto tiene una superficie superior mayor a la real (ver el cubo en la figura 106). El Esto podría sugerir una necesidad de análisis de la imagen en 3D, ya que un área deformada arrojará un centroide

incorrecto. Sin embargo, el sistema no toma en cuenta la posición exacta del centoride para funcionar correctamente. Basta con determinar en qué región de la mesa de trabajo se encuentra (ver figura 108).
El proyecto propone una mesa de trabajo donde los objetos son colocados únicamente por el brazo robot, por lo que se descartan posiciones comprometidas de los objetos donde el error en el centroide cobraría relevancia.

```
%%   Pintar centroides sobre imagen
[centroi_R,centroi_C]=size(centroids);
for i=1:centroi_R
    a=nearest(centroids(i,1));
     b=nearest(centroids(i,2));

     for x=-5:5
    bw(b+x,a)=0;
     end
     for x=-5:5
    bw(b,a+x)=0;
     end
end
```

Figura 107.- Código para colocar las marcas en los centroides

La figura 108 muestra las divisiones de la imagen y los centroides marcados en cada región.

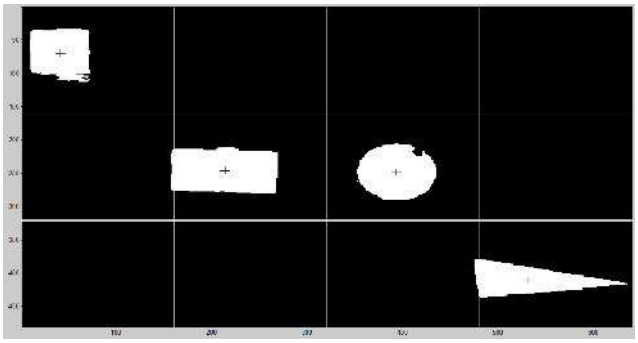

Figura 108.-Imagen con los centroides y divisiones dibujados

Como se mencionó al principio de éste capítulo, la mesa tiene asignados doce lugares para colocar objetos. La utilidad de contar con la información procesada hasta éste momento, es determinar cuáles de los doce lugares están ocupados y cuáles se encuentran libres.

Para ese fin, primero se colocan en un arreglo, las coordenadas de todos los centroides encontrados. El siguiente paso consiste en tomar la posición de cada centroide y determinar en cuál de las 12 regiones de la mesa se encuentra. Estas regiones se pueden consultar en la figura 76.
A continuación se explica el proceso de forma detallada para un centroide muestra:
La figura 109 representa el área de la mesa de trabajo dividida imaginariamente en cuatro franjas.

La medida del largo de dicha mesa está representada por “C”. Consecuentemente, los límites entre las franjas están en intervalos de $(\frac{1}{4})$ C

En la misma figura, también se tiene representado un centroide con coordenadas (a,b). Para determinar suposición en la mesa, se compara su coordenada b contra los límites de las divisiones. Si el número b es mayor que 3C/4, entonces se encuentra en la cuarta región, y ya no se vuelven a hacer comparaciones. En caso contrario, hace la comparación contra el límite inmediato inferior.
Si el número b es mayor que C/2, entonces se encuentra en el tercer región, y ya no se vuelven a hacer comparaciones. Y así, sucesivamente.
Cuando el algoritmo determina la región en la que se encuentra el punto ´b´, guarda ese dato.

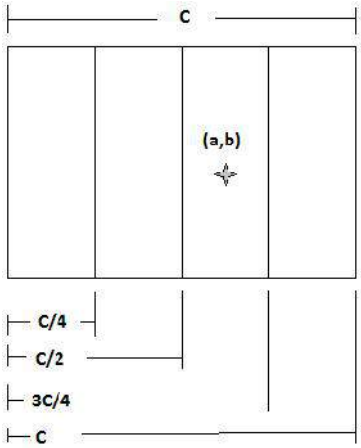

Figura 109.- Algoritmo de búsqueda de lugares en la mesa “X”

La misma lógica se utiliza para la determinación de las regiones en “Y”. En la figura 110, se representa la misma área de la mesa de trabajo; sin embargo, ahora las divisiones imaginarias son horizontales. El algoritmo buscará ahora en qué región se encuentra el punto ´a´. Si el número ‘a’ es mayor que 2R/3, entonces está en la región inferior y ya no hace más comparaciones y se guarda el resultado… Etc.

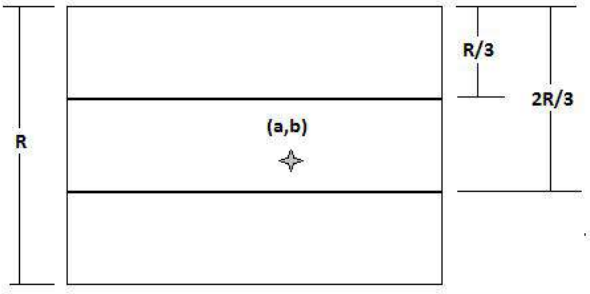

Figura 110.- Algoritmo de búsqueda de lugares en la mesa “Y”

De la misma forma que para las regiones en “X”, se guarda la región en “Y” para cada punto centroide. De manera que para cada centroide se tiene un par de regiones asociadas.

Se toma cada par de regiones y se toma como posiciones de una matriz 3X4.

Inicialmente, dicha matriz es una matriz de “ceros”.
En cada una de éstas posiciones se coloca un “1” lógico. Ésta es la matriz de posiciones, y representa los lugares ocupados en la mesa de trabajo. La figura 111 representa un ejemplo de dicha matriz.

```
areas =

     1     0     1     1
     0     0     0     1
     1     0     1     1
```

Figura 111.- Matriz de posiciones (Existen objetos en las posiciones 1, 3, 4, 8, 9, 11 y 12 de acuerdo a la figura 68)

8.11.3 BLUETOOTH

Si el usuario eligió el modo de detección Bluetooth, se toma la imagen de la mesa de trabajo para determinar los lugares disponibles mediante (Func. 1):

[areas,imagen]=Tomar_foto(0,video_obj);----------(Func. 1)

La imagen resultante, se carga al axes llamado “mapa”. Ver figura 83.

también se carga el dispositivo mediante el siguiente código.

```
B = Bluetooth('HC-05',1,'Timeout',2)
fopen(B);
```

donde: ‘HC-05’.- Es el nombre del dispositivo
‘1’.- Es el canal por donde se leerá el dispositivo
‘2’.-Son los segundos para “Timeout”
fopen(B).- Conecta con el dispositivo.

Luego, constantemente, se busca una lectura de Tag, mediante el llamado constante de la función (2):

[tipo,Lugar]=bluetooth(B)----------(Func. 2)

Cuando un Tag se acerca lo suficiente la unidad lectora de RFID, automáticamente, se comienzan a transmitir las lecturas obtenidas del mismo.
Como resultado de la lectura realizada existen dos posibilidades:

- Lectura Exitosa
- Error

El código de un error de lectura es: `AABB2DEDC`.
Si la lectura fue exitosa, la lectura arrojará: `01FFFFFFFFFFFFFFFFFFFFFFFFFF04`.

La forma en que MATLAB distingue entre estos dos casos, es llevando la cuenta de los Bytes que van llegando.
Cuando la cuenta va en cinco, se leen los Bytes recibidos hasta el momento.
Si los cinco Bytes son: AA `BB 02 DE DC`, Entones se trata de una lectura errónea, y la cuenta se Bytes se reinicia.
Si no recibe lectura dentro del tiempo asignado, se obtiene un '0'.

Al obtener un cero, una lectura exitosa o una lectura errónea, la función termina.

8.11.4 CODIGO QR

Si, por otro lado, el usuario eligió el modo de detección por Código QR, se cargan; la cámara 2 y las librerías de manejo de códigos QR.

Al igual que para bluetooth, también se toma la imagen de la mesa de trabajo para determinar los lugares disponibles mediante la misma (Func. 1):

[areas,imagen]=Tomar_foto(0,video_obj);

La imagen resultante, se carga al axes llamado "mapa". Ver figura 83.

Para leer los códigos QR, se ejecuta continuamente, la (func.3):

[tipo,Lugar]=CodigoQR_Beta(video_2)----------(Func. 3)
La cual entrega un "cero" en la variable "tipo" si no hubo lectura. Esta función no cuenta con mensajes visibles de error.

8.11.5 MANEJO DE LA INFORMACION

Cualquiera de los modos de lectura, arroja dos resultados: "`tipo" y "Lugar"`.

[tipo,Lugar]=CodigoQR_Beta(video_2) ----------(Func. 3)
[tipo,Lugar]=bluetooth(B)----------(Func. 2)

Si la variable "tipo" tiene el valor de cero, significa que no hubo lectura, cualquiera que sea el modo de lectura; y Matlab sigue llamando continuamente a la función.

Si un elemento es identificado, la variable "tipo" ya no será igual a cero.
MATLAB interpretará dicha variable como el tipo de pieza y lo mostrará en el "text1".

El programa tomará entonces una fotografía de la mesa de trabajo, hace una corrección de imagen, identifica objetos situados en la mesa, determina su posición y genera la información de lugares disponibles.
Todo mediante la función (Func. 1):

[matríz,imagen]=Tomar_foto(0,video_obj);----------(Func. 1)

De esta función se obtienen, la matriz deposiciones y la imagen procesada de la mesa de trabajo. Dicha imagen se carga al axes llamado "mapa", y la matriz sirve como argumento para alguna de las siguientes funciones:

[disponible,leyenda]=acomodar(matriz,tipo,check);----------(Func. 4)
[disponible,leyenda]=acomodar3(tipo,Lugar,matriz,check);----------(Func. 5)

Figura 112.- Selección de modalidad del acomodo de piezas

Durante las especificaciones, el usuario seleccionó o dejó sin seleccionar, la opción: "Usar Imagen". Ver figura 112.

De eso depende cuál de las dos funciones anteriores será ejecutada. Se trata de la modalidad del acomodo de piezas, descrito en la parte llamada "Proceso" del presente capítulo.

La función elegida, hará que el robot acomode las piezas en la mesa. Y vuelva a su posición de descanso.
Cada que el robot deje una pieza, se renovará la imagen de la mesa por medio de la función (Func. 1).

Y, finalmente, se mostrará en la interfaz de usuario, la secuencia de movimientos ordenados al robot. Ver figura 117.
Esto último, se obtiene del dato "leyenda "de las funciones (Func. 4) ó (Func. 5), que se genera cuando concatenan las respuestas de cada vez que ejecutan la función (Func. 6) :

[leyenda]=numeroKUKA(numero,enable)----------(Func. 6),
que es llamada varias veces por cualquiera de las funciones antes mencionadas.

Hecho todo esto, se repite la rutina, hasta que el usuario da por terminada la rutina con el botón "Detener".
Cuando esto sucede, se detiene el modulo bluetooth o la lectura QR. Mediante las acciones:

```
fclose(B);
clear('B');

Ó
stop(video_2);
clear video_2;
```

si se trata de modo Bluetooth ó QR, respectivamente.

8.11.6 DETALLE DE LAS FUNCIONES CREADAS

A continuación, se enlistan las funciones creadas para el presente proyecto y se detallan sus parámetros.

function [areas,imagen]=Tomar_foto(mostrar,video_obj)

Esta función toma la imagen de la cámara y la rectifica.

áreas.- Regresa una matriz numérica representando con "0" los espacios vacíos y con "1" los lugares donde hay un bloque. Un ejemplo se muestra en la figura 113.

```
areas =

     1     0     1     1
     0     0     0     1
     1     0     1     1
```

Figura 113.- Matriz de posición de objetos colocados en la mesa de trabajo

Imagen.- Es la imagen tomada de la mesa con los centroides de los objetos encontrados. (Figura 114).

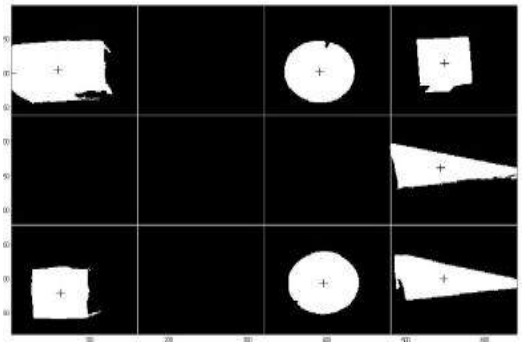

Figura 114.- Imagen de la mesa de trabajo con los centroides de los objetos encontrados

mostrar.-

0.- No muestra la imagen tomada .Figura 114.
1.- Muestra la imagen tomada. Figura 114.

video_obj.- Objeto de video de donde se tomara la imagen.

function [leyenda]=numeroKUKA(numero,enable)

Envía a las salidas del microcontrolador un número de 4 bits, y regresa un string llamado "leyenda", con la acción correspondiente para el robot.

leyenda.- String con la acción correspondiente al número que se dio en la entrada.

numero.- Entero decimal de 0 a 16, que se desea mandar a las entradas digitales del robot.

enable.- '0'.- Simular acción. '1'.- Ejecutar acción

Dentro de ésta función, se Llama a una función disponible previamente llamada "PIC_KUKA('…')", Quien es la encargada de enviar un arreglo binario de 12 elementos al microcontrolador.

Un propósito de la función es Construir el argumento para la función PIC_KUKA(…), y luego ejecutarla **.** Para esto sigue los siguientes pasos:

- Toma un número decimal.
- Lo convierte a binario.
- Invierte el orden de los Bits.
- Lo concatena con otro string de 8 ceros.
- El resultado lo guarda bajo el nombre de "argumento".
- Ejecuta a la función **PIC_KUKA(-argumento-).**

La función se describe gráficamente en la tabla 4. Se muestra las salidas ("Orden al microntrolador" y "Leyenda") producidas por la entrada ("Número").

Numero	Orden al Microcontrolador	Leyenda
0	PIC_KUKA(000000000000)	Descanso
1	PIC_KUKA(000000001000)	Colocar en: 1
2	PIC_KUKA(000000000100)	Colocar en: 2
3	PIC_KUKA(000000001100)	Colocar en: 3
4	PIC_KUKA(000000000010)	Colocar en: 4
5	PIC_KUKA(000000001010)	Colocar en: 5
6	PIC_KUKA(000000000110)	Colocar en: 6
7	PIC_KUKA(000000001110)	Colocar en: 7
8	PIC_KUKA(000000000001)	Colocar en: 8
9	PIC_KUKA(000000001001)	Colocar en: 9
10	PIC_KUKA(000000000101)	Colocar en: 10
11	PIC_KUKA(000000001101)	Colocar en: 11
12	PIC_KUKA(000000000011)	Colocar en: 12
13	PIC_KUKA(000000001011)	Coger Pieza
14	PIC_KUKA(000000000111)	Ajuste D Cámara
15	PIC_KUKA(000000001111)	Regresa Piezas

Tabla 15.- Correspondencia entre los datos de salida, la Función PIC_KUKA, y la leyenda generada

Como se puede observar de la tabla 15, los números binarios, sólo corresponden si se toman los 4 Bits menos significativos y se invierte el orden de los mismos.

function [disponible,leyenda]=acomodar(areas,figura,ejecutar)

La función colocara la pieza en la columna que le corresponda, llenando, en orden, los lugares disponibles. Utiliza la función "numeroKUKA()".

disponible.- Entrega el resultado del análisis de la imagen de la mesa de trabajo

'0'.- No hay lugar para colocar la pieza en la columna que le corresponde .

'1'.- Hay lugar para colocar la pieza en la columna que le corresponde .

leyenda.- String compuesto de las órdenes que se mandaron al robot.

áreas.- Matriz numérica que representa los lugares en la mesa.
A partir de ella, la función decidirá qué lugar debe ser ocupado

figura.- número correspondiente a la figura detectada (ver tabla 16)

Numero	Figura
1	Prisma rectangular
2	Cubo
3	Prisma circular
4	Prisma Triangular

Tabla 16.- Números correspondientes a la figura detectada

ejecutar.-

'0'.- Simular acción.
'1'.- Ejecutar acción

function [disponible,leyenda]=acomodar3(figura,Lugar,areas,ejecutar)

La función acomoda la pieza según el lugar asignado en su identificación capturada. Siempre y cuando el lugar se encuentre disponible. Utiliza la función "numeroKUKA()"

disponible.- Entrega el resultado del análisis de la imagen de la mesa de trabajo

'0'.- No hay lugar para colocar la pieza en la columna que le corresponde .

'1'.- Hay lugar para colocar la pieza en la columna que le corresponde .

leyenda.- String compuesto de las órdenes que se mandaron al robot.

figura.- Número correspondiente a la figura detectada . Tabla 17

Numero	Figura
1	Prisma rectangular
2	Cubo
3	Prisma circular
4	Prisma Triangular

Tabla 17.- Número correspondiente a la figura detectada

Lugar.- Entrada numérico que indica el lugar que debe ocupar el objeto en la mesa

áreas.- Matriz numérica que representa los lugares en la mesa.
A partir de ella, la función decidirá qué lugar debe ser ocupado

ejecutar.-

'0'.- Simular acción.
'1'.- Ejecutar acción.

function [tipo,Lugar]=bluetooth(B)

Función encargada de leer el dato que llega por medio de Bluetooth

tipo.- Dato numérico que identifica el tipo de pieza.

Lugar.- Dato numérico que indica el lugar que tiene asignado la pieza identificada.

B.- Objeto Bluetooth (Las bibliotecas para Bluetooth, están disponibles a partir de la versión R2012b(8.0.0.783) de Matlab.

function [tipo,Lugar]=CodigoQR_Beta(obj2)

Función encargada de leer el dato que llega por medio de la lectura de Código QR

tipo.- Dato numérico que identifica el tipo de pieza.

Lugar.- Dato numérico que indica el lugar que tiene asignado la pieza identificada.

obj2.- objeto de video que contiene el código QR

8.11.6 INTERFAZ DE USUARIO.-

El programa, al correr, tiene la apariencia mostrada en la figura 115. Tiene un menú en la parte superior que pide al usuario elegir entre los dos modos de funcionamiento: QR o RFID.

Se encuentra en la parte inferior, el botón de salir, el cual cerrara la aplicación.
En cuanto el usuario entra al menú, se carga la cámara que está en la mesa de trabajo.
De esta manera, la computadora determina si hay objetos previamente en la mesa y el lugar que ocupan.

Si se elige el modo QR, aparecerá la imagen de la figura 115, donde se aprecian algunas opciones en los checkbox's

Figura 115.- Pantalla de parametros (QR).

La opcion usar imagen permite elegir al usuario entre respetar el lugar grabado en la identidicacion del objeto o, el ir acomodando las piezas en orden de aparicion.

Tambien se puede elegir entre “Simular” o “Ejecutar”. Esto se refiere a las acciones del robot.

- Simular.- Solamente escribir en la interface en forma de texto sin efecto; la orden que el programa le daría al robot (Figura 116).
- Ejecutar.- Mandar la orden a la interfaz del robot para que este realice los movimientos.

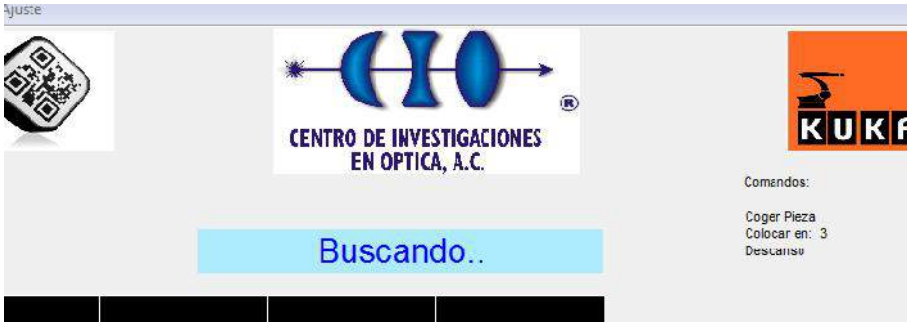

Figura 116.-Mensajes en la interfaz para simulación

Una vez configuradas estas opciones, se da la orden de iniciar la captura de lecturas. La cámara número 1 tomara la imagen para “ver” las piezas que se encuentren en la mesa de trabajo. La apariencia de la aplicación hasta ese momento se ve en la figura 117.

Figura 117.- imagen en la interfaz de lugares ocupados

En caso de haber elegido el modo RFID, la pantalla aparecerá como se muestra en la figura 118. Obsérvese que las opciones de configuración son las mismas.

Figura 118.- Opciones para RFID

Al dar la orden de comenzar, el bluetooth arranca, al igual que la cámara número 1.
La figura muestra que se ha detectado una tarjeta con la información de un círculo. La clase de objeto se muestra en el recuadro azul con texto de la figura 119.

Figura 119.-Determinación de lugares ocupados

Si se desea detener la detección de objetos, solo hay que pulsar el botón “Detener”. También aparecerá esta acción en el mismo recuadro en la figura 120.

Figura 120.- Proceso detenido. Se puede reanudar o salir

Finalmente, La opción “salir”, cerrara la aplicación.

8.12 Resultados

En todas las pruebas realizadas, el sistema funcionó correctamente bajo las dos formas de detección, así como bajo los dos modos de colocación de piezas.

Se probó el sistema arrancando el programa en Matlab, el robot KUKA y conectando la unidad RFID, así como las webcam asociadas y la interfaz USB. Una vez elegido el modo de acomodo, se colocaron los objetos en el punto de identificación.

La identificación de objetos ocurre de forma automática con un éxito del 80% para lecturas de tags RFID en forma de pulsera. Para los Tags tipo tarjeta y llavero, el éxito fue del 100%. Para las lecturas de código QR el éxito fue del 100% .

La detección suele ser más rápida usando el sistema RFID que usando códigos QR. Sin embargo, el uso de éstos tiene la gran ventaja de que no precisan de una distancia rigurosa entre el identificador y el lector para una correcta lectura.

La unidad lectora RFID no presentó falla alguna durante las pruebas. Al presentar un objeto, la lectura se hace inmediatamente y el contenido transmitido vía Bluetooth es interpretado correctamente por la computadora en todas las pruebas efectuadas.

También se colocaron bloques en posiciones aleatorias en la mesa de trabajo antes de colocar bloques en el punto de lectura. Al colocarse los últimos, el sistema logró identificar el lugar que debería ocupar cada bloque e identificar si el espacio se encontraba disponible. Este procedimiento no reportó errores.

Cuando el lugar en la mesa de trabajo se encontraba ocupado, el robot colocaba la pieza en otro lugar acorde al tipo de figura, o bien, no hacía movimiento alguno según la elección del usuario. En ambos casos, la interfaz programada mostró las leyendas correspondientes a los movimientos realizados por el robot.

Con la finalidad de realizar pruebas se programó la opción de Simulación, con la que la interfaz funciona sin mandar órdenes al robot. Bajo ésta modalidad, también se obtuvieron los resultados esperados en el 100% de las pruebas realizadas. El programa muestra solamente las leyendas correspondientes a los movimientos que debió haber hecho el robot.

Los tiempos de respuesta pueden ser reducidos si se toma la imagen de la mesa de trabajo una sola vez antes de comenzar el primer movimiento por parte del robot. De ésta manera, el sistema podrá administrar en memoria, los lugares vacantes sin necesidad de volver a hacer el procesamiento de imagen. La desventaja de hacerlo de esa forma radicaría en que si por alguna razón las piezas no llegasen a la mesa de trabajo, o alguien las quitara, el sistema no tendría forma de registrarlo. En otras palabras, se volvería un sistema de lazo abierto al quedarse sin retroalimentación.

Pueden presentarse problemas al conciliar las dos webcam en MATLAB, ya que a menudo solamente reconoce una de ellas, no obstante el sistema operativo muestre las dos cámaras en el administrador de dispositivos.

EL problema se soluciona volviendo a cargar MATLAB cuidando que las cámaras sigan visibles en el administrador de dispositivos de la PC.

Entre los identificadores RFID, los Tag tipo tarjeta demostraron un mejor desempeño, ya que son detectados mucho más fácilmente que los tipo pulsera y llavero. Adicionalmente, los Tag´s de pulsera, presentan dificultades de lectura si se encuentran flexionados.

8.13 Aplicaciones potenciales

8.13.1 Línea común de paletización y emplayado.

El paletizado o paletización es la acción y efecto de disponer mercancía sobre un palé o tarima para su almacenaje y transporte. La tarima se muestra en la figura 121. Se puede paletizar casi cualquier tipo de mercancía; encima de los palés se suelen colocar cajas y otros embalajes agrupados de forma que se aproveche el espacio del palé y que la carga se mantenga estable. En la figura 122 se muestra un robot paletizando sacos sobre una tarima.

Figura 121.- Pallet o Tarima

Figura 122.- Brazo robótico acomodando productos en un Pallet (Paletización)

Es común en una planta de producción que se tenga varias líneas trabajando al mismo tiempo en productos diferentes. Puede ser que cada línea tenga un robot paletizador, o bien, que una sola estación paletizadora reciba los embalajes de todas las líneas.
Ésta última opción es mucho más conveniente si los diferentes embalajes conservan características comunes. Tales estaciones cuentan con una entrada diferente para cada línea, (Ver figura 123) con lo que el robot debe ir de una entrada a otra tomando paquetes y acomodándolos en los pallets.
La programación de un robot utilizado de esa manera, incluye el ir a cada entrada a recoger la pieza y acomodarla en su lugar.
Para hacer más eficiente el proceso, será importante que si alguna línea no está funcionando, se excluya de la rutina del robot. Entonces, se puede recibir una señal de cada línea activa, y así, el programa decide a que entrada no se acercará.

Figura 123.- Estación de paletización con tres entradas

Cada transportador de las líneas requiere una estructura, rodillos o bandas, motores, rodamientos, y mantenimiento. Todos esos factores representan gastos y probabilidades de falla.
La aplicación propuesta tiene la finalidad de administrar el proceso de colocación de los productos en los pallets. Las líneas de embalaje, deberán llegar a un solo transportador. Como ejemplo ilustrativo, refiérase a la figura 124. El transportador central recibirá los productos que llegan de las diferentes líneas en operación. Y constituirá la única entrada a la estación paletizadora.

Figura 124.- Ejemplo de un trasportador que colecta los productos de varias líneas

Cada embalaje deberá llevar impreso un código QR, de manera que, cuando llegue a la estación de paletización, sea leído por una cámara. La información obtenida indicará al robot cuál de sus rutinas pregrabadas debe ejecutar. Las rutinas pregrabadas deberán contemplar las dimensiones de los objetos, las trayectorias apropiadas y la posición del pallet que le corresponde al tipo de producto.
Consecuentemente, la herramienta en el robot, debe tener un diseño apropiado para sujetar los diferentes tipos de mercancía. En la figura 125 se puede apreciar una herramienta versátil. Adicionalmente, al momento de identificar un embalaje, una cámara puede corroborar la existencia del pallet correspondiente. En caso de no haber, el robot colocará el embalaje fuera del proceso.

Figura 125.- Pinza para varios tamaños de embalaje

Las ventajas de una unidad paletizadora común, basada en la aplicación desarrollada en ésta tesis, consisten en:

- El ahorro en la construcción de transportadores
- Ahorro en el mantenimiento de los mismos
- No será necesario que los embalajes lleguen en orden
- No será necesario que un operador cambie el programa del robot, si alguna o algunas líneas no está trabajando. El funcionamiento será automático
- El sistema de monitorización de pallets evitará que el robot intente colocar mercancía sobre un pallet inexistente. Si esto sucede, el producto sufriría impactos perjudiciales.

8.13.2 Juegos de mesa

"La ingeniería es una de las actividades humanas que ha propiciado la construcción de la infraestructura en la cual se sustenta buena parte del bienestar de la población. En cierta medida, el mundo en que vivimos es producto de la ingeniería […]. De acuerdo con lo anterior, la ingeniería se vincula estrechamente con el proyecto nacional y se constituye en un soporte de las estructuras productivas y de servicios, por lo que representa un factor determinante para el desarrollo económico y social de México" [40].

Salvo en Corea, donde hay un ingeniero cada 625 habitantes y el 25% de los graduados universitarios salen de carreras de Ingeniería, en el resto del mundo los ingenieros son un bien escaso. Los datos dispararon el debate durante el capítulo temático "Formación del ingeniero para el desarrollo sostenible" que se desarrolló en el encuentro Ingeniería 2010 Argentina, Congreso Mundial y Exposición, que sesionó del 17 al 20 de octubre en La Rural (Predio ferial de Buenos Aires, Argentina) [41].

En México de cada 10 niños que ingresan a Primaria, menos de uno se convertirá en Ingeniero señaló el presidente de Grupo Reduca y representante de Vex Robotics en México, Francisco Wilson Robles en el año 2013, quien añadió que los sectores que más resienten la falta de ingenieros son el de la agricultura, la manufactura y el automotriz [42].

A escala nacional, 40 de cada 100 universitarios mexicanos estudian una carrera del área de la ingeniera y, particularmente en Jalisco (la entidad más destacada en la industria electrónica), la cifra no asciende ni al 20%. De los dos millones y medio de estudiantes, la mayoría, opta por carreras de humanidades.

Según el portal CNN Expansión si se siguen atrayendo inversiones extranjeras pero el número de ingenieros locales no crece, el país tendrá profesionales de esa área de origen chino o hindú.

Según una publicación de la red de universidades de habla hispana y portuguesa UNIVERSIA, con fecha 07/03/2013, se tienen las siguientes cifras nacionales [43]:

- 24.000 ingenieros se gradúan al año; una cifra muy baja si se compara con Estados Unidos, donde cada año, egresan 60.000 nuevos profesionales de la ingeniería.
- 24 es el puesto que ocupa México en el mundo por su población de ingenieros. La lista es encabezada por Corea del Sur, Taiwán y Japón.

Para que México cuente con más ingenieros, se les debe hacer entender a los jóvenes que pueden crear, desarrollar prototipos y programarlos, construirlos y manejarlos. Esto les permitirá darse cuenta que pueden hacer cualquier cosa y que el límite es su creatividad [37].
De acuerdo con Monreal Revueltas y Franco Chumillas, entre las soluciones que pueden ser impulsadas para tratar de atraer a los estudiantes hacia el estudio de alguna ingeniería, se pueden destacar [44]:

- Charlas sobre ingeniería.
- Interacción con el alumno.
- Incremento de la enseñanzas prácticas.
- Resolución de problemas de ingeniería.
- Visitas de empresas y de profesionales.
- Proyectos para el diseño de prototipos.

Además de los recursos que acaban de ser comentados, se puede destacar asimismo, la importancia de otras numerosas soluciones que podrían ser igualmente adoptadas, entre las cuales se encuadra el impulso de la colaboración entre institutos de educación secundaria y centros universitarios, el fomento de las actuaciones dirigidas a la formación del profesorado a este respecto y el aumento del nivel de exigencia en el sistema educativo, y la realización de programas y exposiciones de carácter divulgativo.

8.13.2.1 Ajedrez

El proyecto desarrollado en esta tesis es propicio para apoyar el combate a ésta problemática, ya que es susceptible a ser modificado para que el robot industrial KUKA pueda jugar ajedrez contra un humano. Los identificadores de pieza deben ir incrustados en éstas. El robot tendrá el lector junto a su herramienta.
La cámara de monitoreo de área de trabajo, servirá como “el ojo” del sistema. El algoritmo desarrollado identifica la posición de objetos sobre lo que sería el tablero. También detecta espacios vacíos que pueden ser ocupados. Serán necesarios 32 identificadores que “volverán”, es decir, siempre serán los mismos que se utilicen una y otra vez, por lo que no existe inconveniente en utilizar la identificación por RFID. La figura 126 muestra una escena tentativa de la aplicación.

Figura 126.- Robot KUKA moviendo piezas de ajedrez

Al comenzar el juego, el sistema supondrá todas las piezas en su lugar. Cada vez que el jugador humano mueva una pieza, se procesará la imagen de la cámara para encontrar la pieza que se movió. Acto seguido, el robot acercará el lector a dicha pieza para identificarla. La computadora tomará la decisión de como contraatacar y mandará al robot a ejecutar el movimiento. Finalmente, al tomar la pieza, ésta será identificada y se registrará su nueva posición. Cuando el robot finalice el movimiento regresará a su posición y la cámara actualizará la imagen del juego. Cuando una pieza sea eliminada por el jugador, se deberá colocar fuera del tablero y el robot se acercará para identificarla. Por otro lado, las piezas que elimine el robot, serán tomadas, registradas y puestas fuera del tablero por el mismo.

8.13.2.2 Dominó

Otra aplicación es que el robot juegue dominó. En este caso, la cámara, viendo el tablero, dará las posiciones de las fichas, y podrá determinar el lugar que el robot deberá colocar la siguiente. Cada ficha tendrá un Tag con la información de sus dos numeraciones y una señal para que por medio de procesamiento de imagen, se determine su orientación, como se muestra en la figura 127.

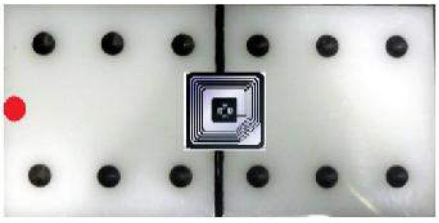

Figura 127.- Propuesta de una ficha de dominó. Incluye un Tag RFID y un punto de referencia para procesamiento de imagen

El robot iniciará el juego con sus fichas ordenadas (con la seña orientada). Al haber un cambio en la imagen obtenida del tablero, el sistema, por medio del procesamiento de imagen dará la posición y el robot se acercará para leer las numeraciones de la ficha. Luego, el sistema determinará cuál de las fichas del robot puede ponerse en el juego. Acto seguido, el robot ejecutará la acción. Las fichas pueden ser Tag´s tipo credencial con impresión de dominó y ser manipuladas por medio de una ventosa.

8.13.3 Farmacia 24 horas 365 días IMSS, ISSSTE, o Seguro Popular.

Debido al incremento del número de derechohabientes a estos servicios médicos, también ha aumentado la demanda de medicamento en sus respectivas farmacias.

No solamente los pacientes que salen de consulta médica necesitan medicamento, también hay personas que tienen que resurtir medicina debido a padecimientos persistentes como la diabetes mellitus, hipertensión arterial, o algún tratamiento de larga duración. Al aumentar la población que demanda el servicio, también aumenta la variedad de situaciones laborales, familiares y personales que hacen que alguna persona no pueda acudir en los horarios de atención y en días hábiles. Un servicio de farmacia que pueda atender las 24 horas los 365 días del año significaría un franco avance en la pobre calidad de la atención actual.

Para la implementación de ésta aplicación es necesaria una reestructuración no muy compleja, aunque tal vez costosa del sistema de farmacias.

Actualmente, en la clínica hospital ISSSTE, las recetas médicas están escritas a mano e incluyen una calcomanía con un código de barras lineal, que contiene información sobre el médico que la emite.
Sera necesario un cambio en las recetas médicas. El médico redactará en una computadora su prescripción. A la hora de imprimir, la computadora incluirá un código QR que contendrá:

- Los medicamentos necesarios
- Su lugar en los estantes de la farmacia,
- Las dosis.
- Y las veces que el paciente puede resurtir cada uno de ellos.

Al salir de consulta, el paciente puede surtir la receta de forma convencional, o bien, ir a la farmacia automatizada, donde no será necesaria la atención de algún empleado. Bastará con acercar el código QR impreso en la receta una videocámara fija empotrada en el muro.

El sistema leerá los medicamentos y los colocará en una lista temporal. Cada medicamento deberá tener una localización fija conocida por el sistema. Un robot de diseño específico parecido al mostrado en la figura 128, recibirá la localización de cada medicamento en la lista para ir por él.

Figura 128.- Robot despachador de medicinas en medio de dos estantes

Adicionalmente, el robot tendrá un pequeño contenedor para ir colocando los medicamentos tomados, y deberá desarrollarse un algoritmo para que la lista de medicamentos se pueda ejecutar en desorden. Esto, debido a que el robot, después de tomar un medicamento, deberá dar preferencia a aquellos de la lista que se encuentren más cercanos a su posición.
Finalmente, extenderá una bandeja a través del muro para que el paciente tome sus medicinas. Deberá existir una base de datos, donde se registre el número de veces que un paciente ha surtido cierta medicina con la misma receta. De esta manera, se evitará que el robot entregue más de lo que el doctor prescribió.

Los registros de medicamentos entregados se llevan a mano, y luego se capturan en el sistema al final del día. El sistema automatizado propuesto hará esto automáticamente y evitará errores humanos en los conteos y captura.

8.13.4 Sistema de recolección de piezas muestra para control de calidad.

Otra aplicación Industrial posible es la de un vehículo recolector de piezas recién maquinadas a lo largo y ancho de una nave industrial para llevarlas automáticamente a los laboratorios de control de calidad.
Las modificaciones necesarias para esta aplicación son más notorias. En primer lugar, no es posible usar un brazo industrial del tamaño del KUKA KR 16-2. Tendrá que ser un sistema robótico más específico, ya que debería ir montado en el vehículo.
Los procedimientos de control de calidad implican el retirar una muestra del producto cada determinado tiempo o cada cierto número de piezas. Dicha muestra es llevada a las áreas destinadas a las mediciones u otras pruebas.

En muchas plantas de manufactura, se realiza la recolección de muestras por mano humana, como se muestra en la figura 129.

Figura 129.- Recolección de piezas muestra por mano humana

La propuesta consiste en que el carrito que se ve en la figura 129, sea autónomo. Tendrá un sistema de auto localización basado en los Tag´s RFID, un robot montado para recoger piezas (Podría ser una pinza suspendida de un puente o un sistema complementario entre el carrito y los contenedores que aguarden la llegada del vehículo) y una cámara web por encima del carrito haciendo la función de la webcam 1 del proyecto de ésta tesis.

Debido a que los horarios de recolección de muestras obedecen a los horarios de trabajo de diferentes máquinas, no se puede tener programado un lugar en la tolva del carrito para cada tipo de pieza. Podrían ser demasiados, y además, no se producen todos los tipos de pieza de forma constante. Por lo que con seguridad, quedarían lugares vacíos en la tolva al mismo tiempo que el algoritmo indica que no se pueden recoger más piezas.

Esta problemática se resuelve tomando una imagen de la tolva con la cámara, para indicar al sistema de lugares vacíos.
La forma de que el carrito siga una ruta establecida será por medio de una línea pintada en el piso. Por lo que se deberá añadir al proyecto un sistema seguidor de línea.
La figura 130 muestra un pequeño prototipo seguidor de línea.

Figura 130.- Prototipo seguidor de línea

Cuando esa línea pase por una estación de maquinado, deberá haber una tarjeta RFID incrustada en el piso. Dicha tarjeta deberá tener grabado el número identificador de la estación. Al mismo tiempo el vehículo autónomo deberá tener el lector RFID debajo de él. Así, cuando pase por ahí, leerá el Tag RFID que está en el piso, y "sabrá" en qué estación se encuentra.

Con ésta información y la rutina programada en él, el sistema a bordo del vehículo determinará si debe detenerse en esa estación, o si debe seguir de paso. En caso de haber ramificaciones en las líneas pintadas en el piso, el sistema de óptico de guía, determinará que se encuentra en "Una bifurcación" y, de acuerdo a la próxima estación en su rutina, "decidirá" hacia dónde debe ir.

Un obstáculo importante es el sistema de control del vehículo. Sería poco práctico llevar una computadora a bordo. Sin embargo, con el rápido avance del desarrollo de dispositivos electrónicos al alcance del público, es posible que pronto se puedan comprar módulos como el YHY502CTG o el HC-05 pero orientados al control de webcams y procesamiento de imágenes. O bien, explorar las posibilidades de que dicho proceso pueda llevarse a cabo mediante una tarjeta electrónica FPGA.
Adicionalmente, se podría evitar el problema si el vehículo emite la información leída de los Tags a un centro de procesamiento vía bluetooth, y reciba la orden de acción por la misma vía.

Capítulo 9

CONCLUSIONES

Del trabajo se concluye que siempre es importante aprovechar el desarrollo de nuevas aplicaciones como los módulos electrónicos disponibles en el mercado. Son fáciles de implementar, ya que están previamente calibrados y acondicionados. Solamente resta el establecer correctamente la comunicación. Desarrollar la aplicación dejó como aprendizaje la familiarización con la programación de un robot industrial, su integración con sistemas aparentemente ajenos, como lo es un microcontrolador o la transmisión inalámbrica de datos.

Otro aspecto del aprendizaje consistió en el manejo de estos módulos comerciales, la existencia de las carpetas SDK, que contienen la información y los controladores para desarrollar aplicaciones propias para el dispositivo.

Por otro lado, podría tenerse la idea de que el protocolo de comunicación RS232, está por desaparecer, y que tal vez no tenga mucho sentido impartirlo en las aulas de ingeniería. Sin embargo, gracias a este proyecto, queda muy en claro que estos conocimientos siguen siendo vigentes y si bien, prácticamente casi han desaparecido de las computadoras personales, siguen siendo muy populares en el ambiente de desarrollo de proyectos a bajo nivel. (Es decir, proyectos basados en microcontroladores).

Una buena razón para utilizar éstos módulos es que son fácilmente integrables, ya que no están acondicionados para una aplicación particular. Esto se hizo evidente durante las primeras etapas del desarrollo del proyecto. Primero se estuvieron haciendo pruebas con un módulo RDW acondicionado. Contaba con carcasa, conexión USB, e interfaz de software para uso inmediato (ver figura 34, Capítulo 4).

Resultó ser más problemático porque los comandos no se manipulaban directamente, sino que se administraban por medio de las funciones de comunicación USB que vienen con el dispositivo (carpeta SDK). Finalmente no fue posible enviar datos hexadecimales por medio de dichas funciones en MATLAB.

Dicho lo anterior se concluye que, en la integración de sistemas es preferible trabajar a bajo nivel aprovechando las ventajas que ofrecen los módulos integrados. Es decir, personalmente, es preferible trabajar con el módulo mostrado en la figura 49 que con el dispositivo de la figura 34.

Sin embargo, con el módulo RDW (Figura 49), se presentó un inconveniente con la comunicación. El comando de lectura (AA BB 0A 21 00 01 FF FF FF FF FF FF 2A) no tiene ningún efecto si se manda directamente como se ve en la figura 60, y la documentación del fabricante no contempla ninguna clase de advertencia o solución. Por intuición propia, se colocaron esperas de 50ms entre cada Byte, obteniendo finalmente el funcionamiento adecuado (Ver figura 3 del apéndice 1. –Líneas 10 a 21).

En el ámbito de un ambiente industrial, es muy útil el contar con una transferencia de datos inalámbrica, ya que sin un tendido de conductores, se contribuye a reducir la condición insegura en la zona de trabajo. Lo que en caso de aplicarse en una nave, llevará a menos accidentes, o bien, a evitar la instalación de líneas de comunicación, mantenimiento de las mismas en caso de una atmósfera agresiva, y su reacomodo en caso de remodelación del lugar de trabajo.

En aplicaciones industriales relacionadas con producto final, puede ser más conveniente usar la identificación por código de barras (QR), ya que la cantidad de embalajes en una línea de producción, podría hacer incosteable el uso de Tag´s RFID pegados en cada uno de ellos.
Las aplicaciones industriales directas no son abundantes, ya que en muchas plantas, las líneas están especializadas en un sólo tipo de producto, y además, con frecuencia las líneas no se encuentran tan cercanas como para implementar el trasportador único propuesto en la figura 125. Por lo que la discriminación al final de la línea, no es algo que pueda implementarse directa e inmediatamente.

La aplicación del vehículo recolector de muestras parece muy interesante y prometedora, ya que aplica a casi todas las empresas en forma casi directa, y con pocas modificaciones para adaptarse a diferentes naves industriales.

Se trata de un proyecto que demuestra la factibilidad de integrar elementos que, por separado parecen ser meras curiosidades. Puede tener seguimiento para poder leer un código de barras lineal con funcionalidades parecidas a las que se tratan en esta tesis. También puede aplicarse en ferias de ingeniería para atraer a los jóvenes a inscribirse a alguna disciplina afín, y contribuir al desarrollo del país al mediano o largo plazo.

Además, la tendencia a la movilidad y la omnipresencia, hacen que cada vez sean más utilizados los sistemas inalámbricos, y el objetivo es ir evitando los cables en todo tipo de comunicación, no solo en el campo de televisión y telefonía, sino en seguridad y domótica. Para este proyecto, se aplicará a sistemas robotizados.

APÉNDICES

Apéndice 1 Programación del microcontrolador y diseño de placa fenólica

Programación del microcontrolador.

Para el control del módulo RWD, se eligió el microcontrolador PIC16F648A por su disponibilidad al momento de realizar el proyecto, por su reducido tamaño y la capacidad de manejar una interrupción externa, además del protocolo de comunicación serial RS-232.

En la figura 2 se muestra el listado de las fuentes de interrupción que soporta el dispositivo.
Para la programación del PIC se eligió el compilador PICC de CCS.

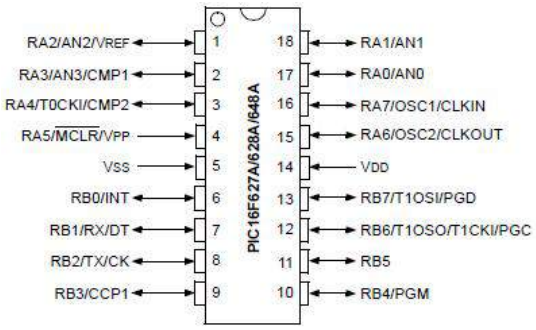

Fig. 1.- PIC16F648A

14.5 Interrupts

The PIC16F627A/628A/648A has 10 sources of interrupt:

- External Interrupt RB0/INT
- TMR0 Overflow Interrupt
- PORTB Change Interrupts (pins RB<7:4>)
- Comparator Interrupt
- USART Interrupt TX
- USART Interrupt RX
- CCP Interrupt
- TMR1 Overflow Interrupt
- TMR2 Match Interrupt
- Data EEPROM Interrupt

Figura 2.- Fuentes de interrupción para el PIC16F648A

El programa, al encender; manda una señal alternada al pin 11, para encender un buzzer indicador de inicio de la rutina (Línea 42). Se definen las variables necesarias, y el programa cae en un ciclo ocioso. (Línea 53-61).
Al ocurrir una interrupción, (transición negativa en pin11 –RB5-), enciende un LED que permanecerá encendido mientras dure la transmisión de datos vía serial. (Línea 6).

PCW

Project Edit Search Options Compile View Tools Debug Document User Tool Bar

Undo Redo Cut Copy Paste Unindent Selection Indent Selection Select All Copy from File Paste to File Playback

Edit

pic_blue.c *

Identifiers

```
#include "E:\Tesis\lector_pic\Auxiliares_pic\comunicacion\pic-bluetooth\pic_blue.h"
int a,b,c,d,e,f,g,h,j,k,l,n,m,indice=0,i;


#int_EXT
EXT_isr()
{
   output_high(pin_B5);     putc(a);
       delay_ms(50);        putc(b);
       delay_ms(50);        putc(c);
       delay_ms(50);        putc(d);
       delay_ms(50);        putc(e);
       delay_ms(50);        putc(f);
       delay_ms(50);        putc(g);
       delay_ms(50);        putc(h);
       delay_ms(50);        putc(j);
       delay_ms(50);        putc(k);
       delay_ms(50);        putc(l);
       delay_ms(50);        putc(m);
       delay_ms(50);        putc(n);
    output_low(pin_B5);

}



void main()
{

   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
   enable_interrupts(INT_EXT);
   enable_interrupts(GLOBAL);
   setup_oscillator(False);
   ext_int_edge( H_TO_L );


////////////////////////////////////////BEEP////////////////////////////////
   output_high(pin_B5);
   delay_ms(100);
   output_low(pin_B5);
   delay_ms(50);
   output_high(pin_B5);
   delay_ms(800);
   output_low(pin_B5);
//////////////////////////////////////////////////////////////////////////////


 a=0XAA;b=0XBB;
c=0X0A; d=0X21; e=0X00; f=0X01; g=0XFF; h=0XFF; j=0XFF; k=0XFF; l=0XFF; m=0XFF;n=0X2A;


while(true)
  {

  }

}
```

63:2 Insert Pjt: pic_blue pic_blue.c

Fig. 3.- Código en PICC para el microcontrolador

A continuación se detallan los fusibles utilizados. En concordancia con los parámetros de la comunicación serial, los baudios se establecen en 19200 (Ver capítulo 7).

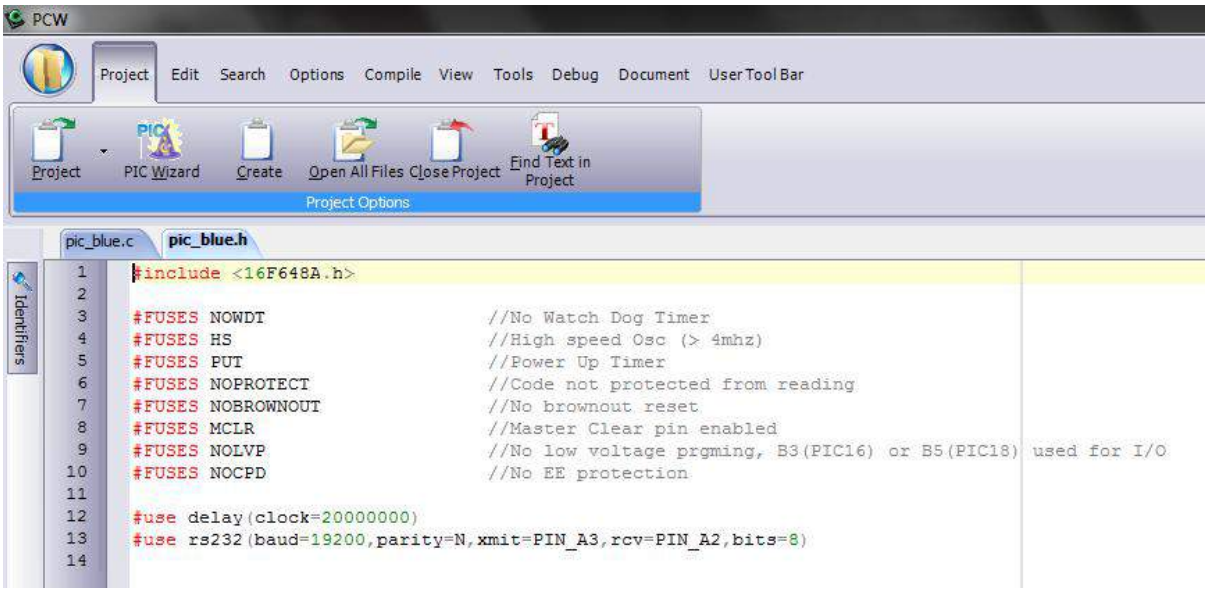

Fig. 4.- Fusibles para el microcontrolador

Diseño de la placa PCB

Para hacer posible de manera estable la interacción entre los módulos y el PIC, se elaboró una placa fenólica, la cual alberga al microcontrolador que gestionará las acciones que deba tomar el módulo RWD, al propio RWD y al módulo transmisor de bluetooth HC-05.

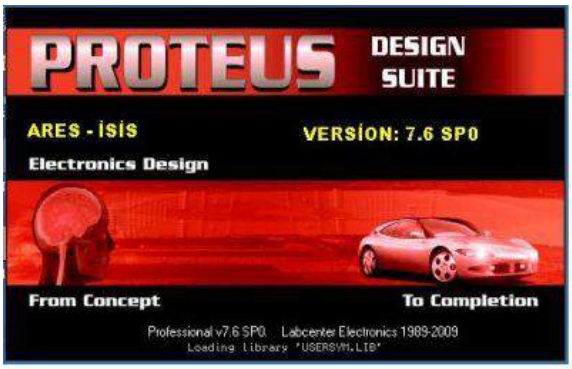

Fig. 5.- Software Proteus

El software usado para la elaboración del diseño electrónico fue Proteus, en sus modalidades "ISIS" y "ARES". El circuito consta de:

- Un pushbutton para la señal de RESET del PIC (RST_PIC).
- Otro pushbutton homologo para el RWD (RST_EUHOYA). Un LED indicador de actividad.
- Un conector hembra (EUHOYAN) de 9 receptáculos. Donde será conectado el RWD.
- Un conector hembra (BLUETOOTH) de 6 receptáculos. Donde será conectado el módulo de Bluetooth HC-05.
- Un transistor de uso común para encender un buzzer de 5V cd (Q1).

El diseño esquemático se muestra en la figura 6.

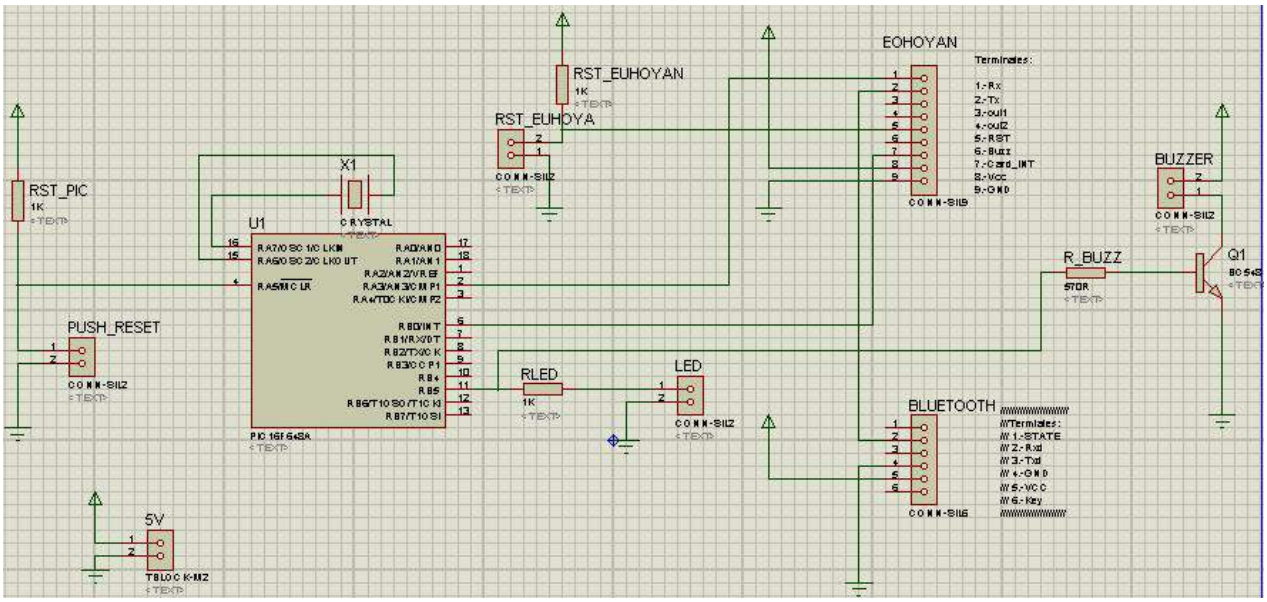

Fig. 6.- Esquemático de la placa

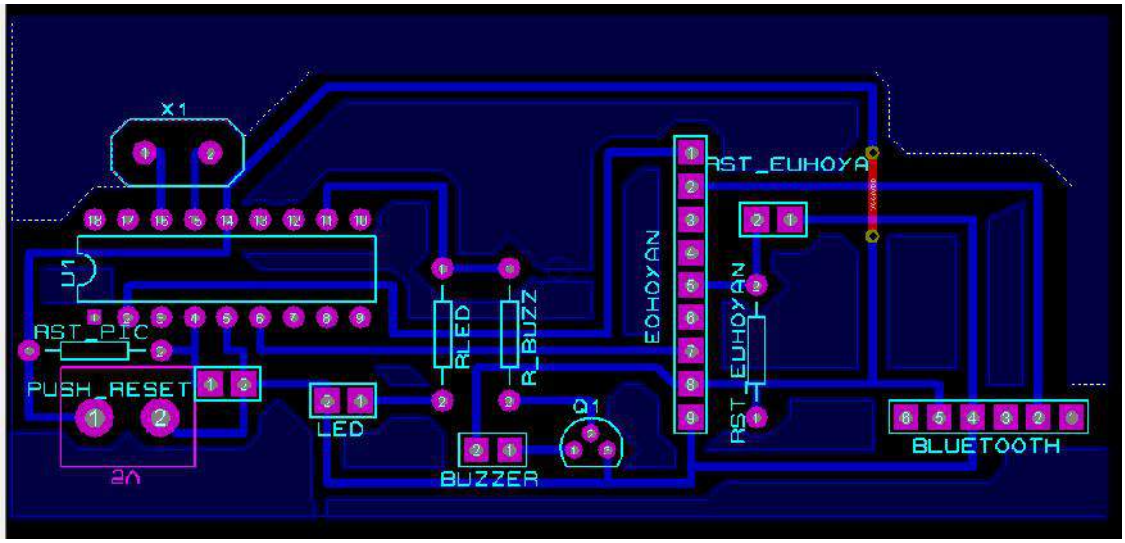

Fig. 7.- Diseño de las pistas

Ares también cuenta con una herramienta que muestra el diseño PCB en 3D. Véanse las figuras 8 y 9.

El circuito construido y montado se muestra en la figura 10,11 y 12. Se pueden apreciar los módulos montados sobre la placa. El módulo RWD en verde, el Modulo HC-05 en azul, y la placa base en color Caqui. Los pushbutton y el LED indicador se encuentran montados sobre la carcasa.

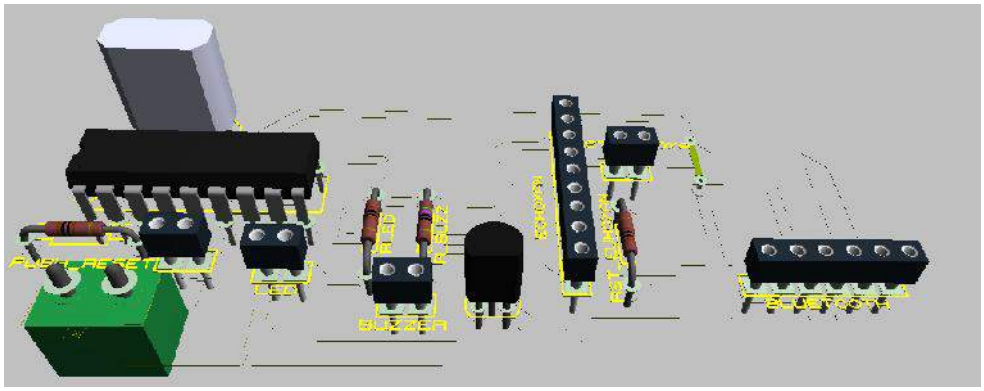

Fig. 8.- Vista superior del diseño del circuito

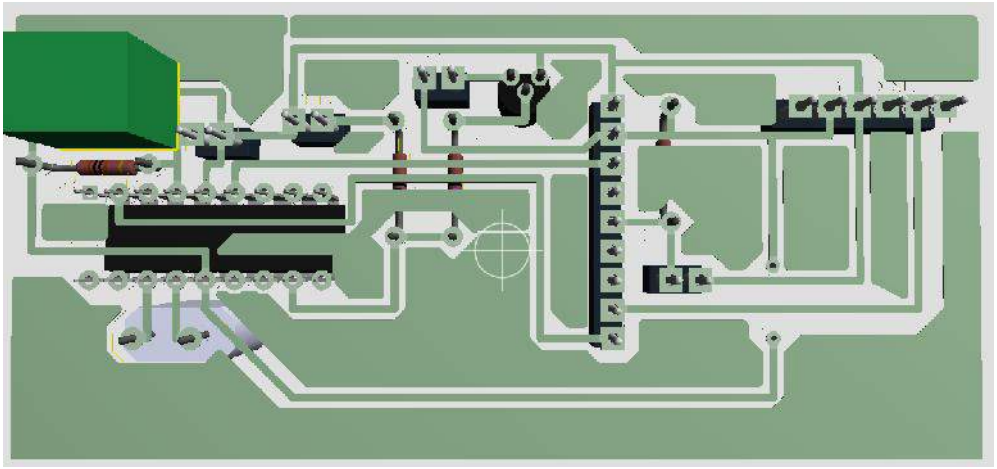

Fig. 9.- Vista inferior del diseño del circuito

Fig. 10.- Integración de la Unidad de lectura

Fig. 11.- Otra vista de la unidad de lectura

Fig. 12.- Unidad de lectura trabajando

Apéndice 2 Webcam1

Fig.1.- Cámara web #1

Especificaciones Técnicas	
Sensor de alta calidad	• Gracias a su sensor SXVGA 1024 x 960 de alta calidad de 1.3 Mega Pixeles, puede obtener imágenes con una resolución de 640 x 480 pixeles.
Night Vision	• Mejora la calidad de las imágenes en cualquier situación de luz.
Diseño moderno	• El diseño de esta cámara web se adapta a cualquier estilo de computadora.
Base tripié	• Cuenta con una base de tripié con la que podrás girar la cámara en cualquier ángulo.
Interfaz	• USB 1.1/2.0
Frecuencia de cuadro	• De 15 hasta 30 fps (cuadros por segundo).
Micrófono	• Si
Enfoque manual	• Para mayor comodidad y exactitud, puede enfocar manualmente su cámara girando la lente de la misma.

Requisitos del sistema
• Compatible con sistemas operativos Windows 98SE/ME/2000/XP/7 • Requiere puerto USB • Unidad de CD-ROM para la instalación del driver

Apéndice 3 Webcam 2

Figura 2 .- Webcam2.

Fabricante: Logitech

Código de Fabricante: 960-000540

- Nombre de Marca: Logitech
- Línea de Producto: QuickCam
- Modelo de Producto: C120
- Tipo de Producto: Cámara Web
- Pantalla y gráficos
- Resolución efectiva: 300 Kilopixel
- Resolución de Video: 640 x 480 @ 30 fps
- Sensor de Imagen: CMOS
- Interfaces/Puertos
- Interfaces/Puertos: 1 x USB 2.0 USB
- Características Físicas
- Color: Black
- Varios
- Sistema Soportado: PC

Apéndice 4 Maxi Code

El código Maxi (figura 1) es un sistema de codificación de datos para visión artificial de dominio público, creado y utilizado originalmente por **UPS Inc**., una compañía de servicio de paquetería fundada en Seattle, Washington ,en 1907 cuyo logo se muestra en la figura 2.

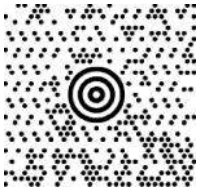

Figura 1.- Código MAXI

Figura 2.- UPS Inc.

Es muy parecido a un código de barras, pero usa puntos acomodados en patrones hexagonales, en lugar de líneas. Está regido por la norma ISO/IEC 16023. Consta de un cuadro de una pulgada de arista, con un ojo de buey en la parte central, rodeado de puntos.
Puede almacenar hasta 93 caracteres, y se pueden concatenar hasta más de 8 imágenes para representar una cantidad mayor de datos. El ojo de buey es muy útil para la localización a través de un sistema de visión, sin importar la orientación, y permite una lectura incluso en movimiento.

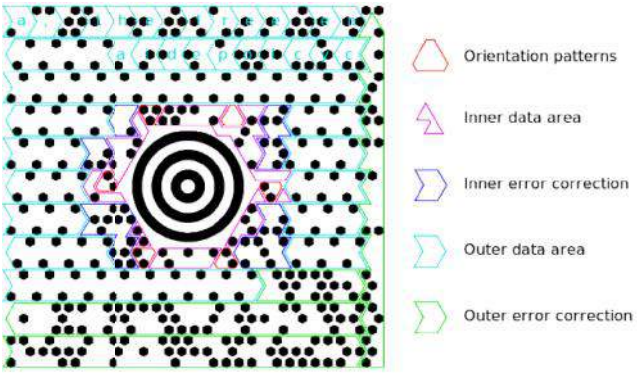

Figura 3.- Parámetros del código Maxi

La información almacenada en MaxiCode, está protegida con un sistema de corrección de errores tal que pueden recuperarse datos en caso de que la imagen haya sido dañada.
El sistema de corrección de errores incluye:

- Indicación del modo en 4bits (comúnmente 2 o 3).
- Código postal.
- Código de país en 3 dígitos bajo la norma SO 3166.
- Código de servicio en 3 dígitos asignado por la compañía.

La porción estructurada del mensaje, se encuentra en el área interior, cerca del ojo de buey (figura 3).
En algunos modos que no utilizan porción estructurada, dicha área solo forma parte del mensaje.

Apéndice 5 Comandos del módulo RWD

Commands & Respones

5.1 Module Type

Command description: Read module type

Data Frame Format:

Send	Head	Length	Command	XOR Checksum
	AA BB	02	01	03

Receive	Head	Length	Command	Module Type	XOR Checksum
Success	AA BB	0A	01	8Bytes	XOR Checksum
Failure	AA BB	02	FE	-	FC

Example:

Send	**AA BB 02 01 *03***	
Description	AA BB	*Head of this COMMAND*
	02	*Length of this COMMAND*
	01	*COMMAND*
	03	*02⊕01*
Receive(Success)	**AA BB 0A 01 *48 59 35 30 32 43 20 20 6E***	
Description	AA BB	*Head of this DATA*
	0A	*Length of this DATA*
	01	*COMMAND*
	48 59 35 30 32 43 20 20	*Module TYPE*
	6E	*0A⊕01⊕48⊕59⊕35⊕30⊕32⊕43⊕20⊕20*
Receive(Failure)	**AA BB 02 FE *FC***	
Description	AA BB	*Head of this DATA*
	02	*Length of this DATA*
	FE	*One's complement of COMMAND*
	FC	*02⊕FE*

5.2 Module Serial Number

Command description: Read Module Serial Number

Note: Each module has it's unique serial number.(NOT card serial number)

Data Frame Format:

Send	Head	Length	Command	XOR Checksum
	AA BB	02	02	00

Receive	Head	Length	Command	Module SN	XOR Checksum
Success	AA BB	06	02	4Bytes	XOR Checksum
Failure	AA BB	02	FD	-	FF

Example:

Send	**AA BB 02 02 *00***	
Description	AA BB	*Head of this COMMAND*
	02	*Length of this COMMAND*
	02	*COMMAND*

	00	02 ⊕02
Receive(Success)	**AA BB 06 02 00 00 00 01 05**	
Description	AA BB	Head of this DATA
	06	Length of this DATA
	02	COMMAND
	00 00 00 01	Module SN
	05	06 ⊕02 ⊕00 ⊕00 ⊕00 ⊕01
Receive(Failure)	**AA BB 02 FD FF**	
Description	AA BB	Head of this DATA
	02	Length of this DATA
	FD	One's complement of COMMAND
	FF	02 ⊕FD

5.3 Power Down

Command description: After execute this Command the module will power down , To wake up the module need to give the RST pin a low-level pulse or Re-power on.

Data Frame Format:

Send	Head	Length	Command	XOR Checksum
	AA BB	02	03	01

Receive	Head	Length	Command	XOR Checksum
Success	AA BB	02	03	01
Failure	AA BB	02	FC	FE

Example:

Send	**AA BB 02 03 01**	
Description	AA BB	Head of this COMMAND
	02	Length of this COMMAND
	03	COMMAND
	01	02 ⊕03
Receive(Success)	**AA BB 02 03 01**	
Description	AA BB	Head of this DATA
	02	Length of this DATA
	03	COMMAND
	01	02 ⊕03
Receive(Failure)	**AA BB 02 FC FE**	
Description	AA BB	Head of this DATA
	02	Length of this DATA
	FC	One's complement of COMMAND
	FE	02 ⊕FC

5.4 Module Firmware Version

Command description: Read Module Firmware Version

Data Frame Format:

Send	Head	Length	Command	XOR Checksum
	AA BB	02	10	12

Receive	Head	Length	Command	Module Firmware Version	XOR Checksum
Success	AA BB	06	10	4Bytes	XOR Checksum
Failure	AA BB	02	EF	-	ED

Example:

Send	**AA BB 02 10 *12***	
Description	AA BB	*Head of this COMMAND*
	02	*Length of this COMMAND*
	10	*COMMAND*
	12	*02 ⊕ 10*
Receive(Success)	**AA BB 06 10 *00 00 02 01* *15***	
Description	AA BB	*Head of this DATA*
	06	*Length of this DATA*
	10	*COMMAND*
	00 00 02 01	*Module SN*
	15	*06 ⊕ 10 ⊕ 00 ⊕ 00 ⊕ 02 ⊕ 01*
Receive(Failure)	**AA BB 02 EF *ED***	
Description	AA BB	*Head of this DATA*
	02	*Length of this DATA*
	EF	*One's complement of COMMAND*
	ED	*02 ⊕ EF*

5.5 Antenna control

Command description: Set the Module antenna power on or off .This command will switch RF field.

Data Frame Format:

Send	Head	Length	Command	Data	XOR Checksum
	AA BB	03	11	1Byte '00': antenna off '01': antenna on	XOR Checksum

Receive	Head	Length	Command	XOR Checksum
Success	AA BB	02	11	13
Failure	AA BB	02	EE	EC

Example:

Send	**AA BB 03 11 00** ***12***	
Description	AA BB	*Head of this COMMAND*
	03	*Length of this COMMAND*
	11	*COMMAND*
	00	*00:* antenna off
	12	*03 ⊕11 ⊕00*
Receive(Success)	**AA BB 02 11** ***13***	
Description	AA BB	*Head of this DATA*
	02	*Length of this DATA*
	11	*COMMAND*
	13	*02 ⊕11*
Receive(Failure)	**AA BB 02 EE** ***EC***	
Description	AA BB	*Head of this DATA*
	02	*Length of this DATA*
	EE	*One's complement of COMMAND*
	EC	*02 ⊕EE*

5.6 Card IDLE

Command description: Set the Card into IDLE . After successfully operation the card will be idle. Reactivate the card need to remove the card from antenna area and put the card into antenna area again.

Data Frame Format:

Send	Head	Length	Command	XOR Checksum
	AA BB	02	12	10

Receive	Head	Length	Command	XOR Checksum
Success	AA BB	02	12	10
Failure	AA BB	02	ED	EF

Example:

Send	**AA BB 02 12 *10***	
Description	AA BB	*Head of this COMMAND*
	02	*Length of this COMMAND*
	12	*COMMAND*
	10	*02 ⊕12*
Receive(Success)	**AA BB 02 12 *10***	
Description	AA BB	*Head of this DATA*
	02	*Length of this DATA*
	12	*COMMAND*
	10	*02 ⊕12*
Receive(Failure)	**AA BB 02 ED *EF***	
Description	AA BB	*Head of this DATA*
	02	*Length of this DATA*
	ED	*One's complement of COMMAND*
	EF	*02 ⊕ED*

5.7 Seek

Command description: Set the module automatic search cards, 1 byte of data. 0x01 open automatic search cards, 0x00 closed. SIG pin active low when find a card untill remove the card or card idle.

Data Frame Format:

Send	Head	Length	Command	Data	XOR Checksum
	AA BB	03	13	1Byte '01': seek on '00': seek off	XOR Checksum

Receive	Head	Length	Command	XOR Checksum
Success	AA BB	02	13	11
Failure	AA BB	02	EC	EE

Example:

Send	**AA BB 03 13 00 *10***	
Description	AA BB	*Head of this COMMAND*
	03	*Length of this COMMAND*
	13	*COMMAND*
	00	*00: auto off*
	10	*03 ⊕13 ⊕00*
Receive(Success)	**AA BB 02 13 *11***	

Description	AA BB	*Head of this DATA*
	02	*Length of this DATA*
	13	*COMMAND*
	11	*02 ⊕ 13*
Receive(Failure)	**AA BB 02 EC *EE***	
Description	AA BB	*Head of this DATA*
	02	*Length of this DATA*
	EC	*One's complement of COMMAND*
	EE	*02 ⊕ EC*

5.8 Set Buzzer ON/OFF

Command description: Set the buzzer ON or OFF, and control the buzzer beep times.

Data Frame Format:

Send	Head	Length	Command	Data	XOR Checksum
	AA BB	03	14	1Byte *'1y'*: Buzzer ON and sound *y* times *'0F'*: Buzzer OFF	XOR Checksum

Receive	Head	Length	Command	XOR Checksum
Success	AA BB	02	14	16
Failure	AA BB	02	EB	E9

Example:

Send	**AA BB 03 14 13 *04***	
Description	AA BB	*Head of this COMMAND*
	03	*Length of this COMMAND*
	14	*COMMAND*
	13	*beep 3 times*
	04	*03 ⊕ 14 ⊕ 13*
Receive(Success)	**AA BB 02 14 *16***	
Description	AA BB	*Head of this DATA*
	02	*Length of this DATA*
	14	*COMMAND*
	16	*02 ⊕ 14*
Receive(Failure)	**AA BB 02 EB *E9***	
Description	AA BB	*Head of this DATA*
	02	*Length of this DATA*
	EB	*One's complement of COMMAND*
	E9	*02 ⊕ EB*

5.9 Set buzzer beep time interval

Command description: Set buzzer beep time interval .

Data Frame Format:

Send	Head	Length	Command	Ringing Interval	XOR Checksum
	AA BB	03	15	1Byte	XOR Checksum

Receive	Head	Length	Command	XOR Checksum
Success	AA BB	02	15	17
Failure	AA BB	02	EA	E8

Example:

Send	AA BB 03 15 10 06	
Description	AA BB	Head of this COMMAND
	03	Length of this COMMAND
	15	COMMAND
	10	Beep time Interval
	06	03⊕15⊕10
Receive(Success)	AA BB 02 15 17	
Description	AA BB	Head of this DATA
	02	Length of this DATA
	15	COMMAND
	17	02⊕17
Receive(Failure)	AA BB 02 EA E8	
Description	AA BB	Head of this DATA
	02	Length of this DATA
	EA	One's complement of COMMAND
	E8	02⊕EA

5.10 Output 1

Command description: Set Output1

Data Frame Format:

Send	Head	Length	Command	Data	XOR Checksum
	AA BB	03	16	1Byte '00': Output '0' '01': Output '1'	XOR Checksum

Receive	Head	Length	Command	XOR Checksum
Success	AA BB	02	16	14
Failure	AA BB	02	E9	EB

Example:

Send	AA BB 03 16 01 04	
Description	AA BB	Head of this COMMAND
	03	Length of this COMMAND
	16	COMMAND
	01	Output 1
	04	03⊕16⊕01
Receive(Success)	AA BB 02 16 14	
Description	AA BB	Head of this DATA
	02	Length of this DATA
	15	COMMAND
	17	02⊕17
Receive(Failure)	AA BB 02 E9 EB	
Description	AA BB	Head of this DATA
	02	Length of this DATA
	E9	One's complement of COMMAND
	EB	02⊕E9

5.11 Output 2

Command description: Set Output2

Data Frame Format:

Send	Head	Length	Command	Data	XOR Checksum

	AA BB	03	17	1Byte '00': Output '0' '01': Output '1'	XOR Checksum

Receive	Head	Length	Command	XOR Checksum
Success	AA BB	02	17	15
Failure	AA BB	02	E8	EA

Example:

Send	**AA BB 03 17 01 *05***	
Description	AA BB 03 17 01 *05*	*Head of this COMMAND* *Length of this COMMAND* *COMMAND* *Output 1* *03 ⊕ 17 ⊕ 01*
Receive(Success)	**AA BB 02 17 *15***	
Description	AA BB 02 17 *15*	*Head of this DATA* *Length of this DATA* *COMMAND* *02 ⊕ 17*
Receive(Failure)	**AA BB 02 E8 *EA***	
Description	AA BB 02 E8 *EA*	*Head of this DATA* *Length of this DATA* *One's complement of COMMAND* *02 ⊕ E8*

5.12 Card Type

Command description: Read card type. S50 card is *'0x0400'*, S70 card is *'0x0200'*, the others can refer to card datasheet.

Data Frame Format:

Send	Head	Length	Command	XOR Checksum
	AA BB	02	19	1B

Receive	Head	Length	Command	Card Type	XOR Checksum
Success	AA BB	04	19	2Bytes	XOR Checksum
Failure	AA BB	02	E6	-	E4

Example:

Send	**AA BB 02 19 *1B***	
Description	AA BB 02 19 *1B*	*Head of this COMMAND* *Length of this COMMAND* *COMMAND* *02 ⊕ 19*
Receive(Success)	**AA BB 04 19 04 00 *19***	
Description	AA BB 04 19 04 00 *19*	*Head of this DATA* *Length of this DATA* *COMMAND* *Card TYPE 04 00: S50 Card; 02 00: S70 Card* *02 ⊕ 19 ⊕ 04 ⊕ 00*
Receive(Failure)	**AA BB 02 E6 *E4***	
Description	AA BB 02	*Head of this DATA* *Length of this DATA*
	E6 *E4*	*One's complement of COMMAND* *02 ⊕ E6*

5.13 Card serial number

Command description: This command reads card serial number

Data Frame Format:

Send	Head	Length	Command	XOR Checksum
	AA BB	02	20	22

Receive	Head	Length	Command	Card SN	XOR Checksum
Success	AA BB	06	20	4Bytes	XOR Checksum
Failure	AA BB	02	DF	-	DD

Example:

Send	**AA BB 02 20 *22***	
Description	AA BB	*Head of this COMMAND*
	02	*Length of this COMMAND*
	20	*COMMAND*
	22	*02 ⊕ 20*
Receive(Success)	**AA BB 06 20 *92 BF 72 59* *20***	
Description	AA BB	*Head of this DATA*
	06	*Length of this DATA*
	20	*COMMAND*
	92 BF 72 59	*Card SN*
	20	*06 ⊕ 20 ⊕ 92 ⊕ BF ⊕ 72 ⊕ 59*
Receive(Failure)	**AA BB 02 DF *DD***	
Description	AA BB	*Head of this DATA*
	02	*Length of this DATA*
	DF	*One's complement of COMMAND*
	DD	*02 ⊕ DF*

5.14 Block read

Command description: Read data from appointed card's block.

Data Frame Format:

Send	Head	Length	Command	Key A or Key B	Block Number	Key	XOR Checksum
	AA BB	0A	21	1Byte '00': Key A '01': Key B	1Byte	6 Bytes	XOR Checksum

Receive	Head	Length	Command	Block Data	XOR Checksum
Success	AA BB	12	21	16Bytes	XOR Checksum
Failure	AA BB	02	DE	-	DC

Example:

Send	AA BB 0A 21 00 08 FF FF FF FF FF FF 23	
Description	AA BB	Head of this COMMAND
	0A	Length of this COMMAND
	21	COMMAND
	00	Authenticate with A Key
	08	Read Block 08
	FF FF FF FF FF FF	Keys
	23	0A ⊕21 ⊕00 ⊕08 ⊕FF ⊕FF ⊕FF ⊕FF ⊕FF ⊕FF
Receive(Success) (*)	AA BB 12 21 00 11 22 33 44 55 66 77 88 99 AA 00 BB CC DD EE FF 23	
Description	AA BB	Head of this DATA
	12	Length of this DATA
	21	COMMAND
	00 11 22 33 44 55 66 77 88 99 AA 00 BB CC DD EE FF	16 Bytes Data of Block 08
	23	12 ⊕21 ⊕00 ⊕11 ⊕22 ⊕33 ⊕44 ⊕55 ⊕66 ⊕77 ⊕88 ⊕99 ⊕AA ⊕BB ⊕CC ⊕DD ⊕EE ⊕FF
Receive(Failure)	AA BB 02 DE DC	
Description	AA BB	Head of this DATA
	02	Length of this DATA
	DE	One's complement of COMMAND
	DC	02 ⊕DE

(*): 00 is added but the length does not change.

5.15 Block Write

Command description: Write data to appointed card's block.

Data Frame Format:

Send	Head	Length	Command	Key A or Key B	Block Number	Key	Data want to write	XOR Checksun
	AA BB	1A	22	1Byte '00': Key A '01': Key B	1Byte	6 Bytes	16Bytes	XOR Checksun

Receive	Head	Length	Command	XOR Checksum
Success	AA BB	02	22	20
Failure	AA BB	02	DD	DF

Example:

Send (*)	**AA BB 1A 22 00 08 FF FF FF FF FF FF 00 11 22 33 44 55 66 77 88 99 AA 00 BB CC DD EE FF 30**	
Description	AA BB 1A 22 00 08 FF FF FF FF FF FF 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 30	*Head of this COMMAND* *Length of this COMMAND* *COMMAND* *Authenticate with A Key* *Read Block 08* *Keys* *16 Bytes Data want to Write* *1A ⊕21 ⊕00 ⊕08 ⊕FF ⊕FF ⊕FF ⊕FF ⊕FF ⊕00 ⊕11 ⊕22 ⊕33 ⊕44 ⊕55 ⊕66 ⊕77 ⊕88 ⊕99 ⊕AA ⊕BB ⊕CC ⊕DD ⊕EE ⊕FF*
Receive(Success)	**AA BB 02 22 20**	
Description	AA BB 02 22 20	*Head of this DATA* *Length of this DATA* *COMMAND* *02 ⊕22*
Receive(Failure)	**AA BB 02 DD DF**	
Description	AA BB 02 DD	*Head of this DATA* *Length of this DATA* *One's complement of COMMAND*
	DF	*02 ⊕DD*

(*): 00 is added but the length does not change.

5.16 Initialize ePurse

Command description: Initialize block as epurse value, the 4-byte purse value of command related to purse operation is low byte first, and the purse value is 4 bytes signed.

Data Frame Format:

Send	Head	Length	Command	Key A or Key B	Block Number	Key	Purse Value	XOR Checksum
	AA BB	0E	23	1Byte '00': Key A '01': Key B	1Byte	6 Bytes	4Bytes (LSB…MSB)	XOR Checksum

Receive	Head	Length	Command	XOR Checksum
Success	AA BB	02	23	21
Failure	AA BB	02	DC	DE

Example:

Send	AA BB 0E 23 00 09 FF FF FF FF FF FF 11 11 00 00 24	
Description	AA BB	Head of this COMMAND
	0E	Length of this COMMAND
	23	COMMAND
	00	Authenticate with A Key
	09	Initialize Block 09 as a Purse
	FF FF FF FF FF FF	Keys
	11 11 00 00	4 Bytes Value of Purse
	24	0E ⊕23 ⊕00 ⊕09 ⊕FF ⊕FF ⊕FF ⊕FF ⊕FF ⊕FF ⊕11 ⊕11 ⊕00 ⊕00
Receive(Success)	AA BB 02 23 21	
Description	AA BB	Head of this DATA
	02	Length of this DATA
	23	COMMAND
	21	02 ⊕23
Receive(Failure)	AA BB 02 DC DE	
Description	AA BB	Head of this DATA
	02	Length of this DATA
	DC	One's complement of COMMAND
	DE	02 ⊕DC

5.17 Read Purse Value

Command description: Read purse value.

Data Frame Format:

Send	Head	Length	Command	Key A or Key B	Block Number	Key	XOR Checksum
	AA BB	0A	24	1Byte '00': Key A '01': Key B	1Byte	6Bytes	XOR Checksum

Receive	Head	Length	Command	Purse Value	XOR Checksum
Success	AA BB	06	24	4Bytes (LSB…MSB)	XOR Checksum
Failure	AA BB	02	DB	-	D9

Example:

Send	AA BB 0A 24 00 09 FF FF FF FF FF FF 27	
Description	AA BB	Head of this COMMAND
	0A	Length of this COMMAND
	24	COMMAND
	00	Authenticate with A Key
	09	Block 09 is a Purse
	FF FF FF FF FF FF	Keys
	27	0A ⊕24 ⊕00 ⊕09 ⊕FF ⊕FF ⊕FF ⊕FF ⊕FF ⊕FF
Receive(Success)	AA BB 06 24 11 11 00 00 22	
Description	AA BB	Head of this DATA
	06	Length of this DATA
	24	COMMAND
	11 11 00 00	Value of Purse
	22	06 ⊕24 ⊕11 ⊕11 ⊕00 ⊕00
Receive(Failure)	AA BB 02 DB D9	
Description	AA BB	Head of this DATA
	02	Length of this DATA
	DB	One's complement of COMMAND
	D9	02 ⊕DB

5.18 Increase Purse Value

Command description: Increase purse value

Data Frame Format:

Send	Head	Length	Command	Key A or Key B	Block Number	Key	Increase Value	XOR Checksu
	AA BB	0E	25	1Byte '00': Key A '01': Key B	1Byte	6 Bytes	4Bytes (LSB…MSB)	XOR Checksu

Receive	Head	Length	Command	XOR Checksum
Success	AA BB	02	25	27
Failure	AA BB	02	DA	D8

Example:

Send	AA BB 0E 25 00 09 FF FF FF FF FF FF 11 11 00 00 22	
Description	AA BB	Head of this COMMAND
	0E	Length of this COMMAND
	25	COMMAND
	00	Authenticate with A Key
	09	Block 09 is a Purse
	FF FF FF FF FF FF	Keys
	11 11 00 00	Vlaue of Increase
	22	0E ⊕25 ⊕00 ⊕09 ⊕FF ⊕FF ⊕FF ⊕FF ⊕FF ⊕11 ⊕11 ⊕00 ⊕00
Receive(Success)	AA BB 02 25 27	
Description	AA BB	Head of this DATA
	02	Length of this DATA
	25	COMMAND
	27	02 ⊕25
Receive(Failure)	AA BB 02 DA D8	
Description	AA BB	Head of this DATA
	02	Length of this DATA
	DA	One's complement of COMMAND
	D8	02 ⊕DA

5.19 Decrease Purse Value

Command description: Decrease purse value

Data Frame Format:

Send	Head	Length	Command	Key A or Key B	Block Number	Key	Decrease Value	XOR Checksum
	AA BB	0E	26	1Byte '00': Key A '01': Key B	1Byte	6 Bytes	4Bytes (LSB…MSB)	XOR Checksum

Receive	Head	Length	Command	XOR Checksum
Success	AA BB	02	26	24
Failure	AA BB	02	D9	DB

Example:

Send	AA BB 0E 26 00 09 FF FF FF FF FF FF 11 11 00 00 21	
Description	AA BB	Head of this COMMAND
	0E	Length of this COMMAND
	26	COMMAND
	00	Authenticate with A Key
	09	Block 09 is a Purse
	FF FF FF FF FF FF	Keys
	11 11 00 00	Vlaue of Decrease
	21	0E ⊕26 ⊕00 ⊕09 ⊕FF ⊕FF ⊕FF ⊕FF ⊕FF ⊕FF ⊕11 ⊕11 ⊕00 ⊕00
Receive(Success)	AA BB 02 26 24	
Description	AA BB	Head of this DATA
	02	Length of this DATA
	26	COMMAND
	24	02 ⊕26
Receive(Failure)	AA BB 02 D9 DB	
Description	AA BB	Head of this DATA
	02	Length of this DATA
	D9	One's complement of COMMAND
	DB	02 ⊕D9

5.20 Read Module's EEPROM

Command description: Read data from module's EEPROM, this module has 16 bytes eeprom to read and write.

Data Frame Format:

Send	Head	Length	Command	Block Num	XOR Checksum
	AA BB	03	32	1Byte	XOR Checksum

Receive	Head	Length	Command	EEPROM Data	XOR Checksum
Success	AA BB	18	32	16 Bytes	XOR Checksum
Failure	AA BB	02	CD	-	CF

Example:

Send	AA BB 03 32 00 31

Description	AA BB 03 32 00 31	Head of this COMMAND Length of this COMMAND COMMAND The first block EEPROM Address MSB 03 ⊕ 32 ⊕ 00
Receive(Success)	**AA BB 12 32 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 20**	
Description	AA BB 12 32 FF..FF 20	Head of this DATA Length of this DATA COMMAND EEPROM Data 12 ⊕ 32
Receive(Failure)	**AA BB 02 CD CF**	
Description	AA BB 02 CD CF	Head of this DATA Length of this DATA One's complement of COMMAND 02 ⊕ CD

5.21 Write Module's EEPROM

Command description: Write data to module's EEPROM, this module has 16 bytes eeprom to read and write.

Data Frame Format:

Send	Head	Length	Command	Block Num	Data	XOR Checksum
	AA BB	13	33	1Byte	16 Bytes	XOR Checksum

Receive	Head	Length	Command	XOR Checksum
Success	AA BB	02	33	31
Failure	AA BB	02	CC	CE

Example:

Send	**AA BB 13 33 00 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 20**	
Description	AA BB 13 33 00 00 .. 0f 20	Head of this COMMAND Length of this COMMAND COMMAND EEPROM block 0 Data to be Writed 13 ⊕ 33 ⊕ 00.. ⊕ 0f
Receive(Success)	**AA BB 02 33 31**	
Description	AA BB 02 33 31	Head of this DATA Length of this DATA COMMAND 02 ⊕ 33
Receive(Failure)	**AA BB 02 CC CE**	
Description	AA BB 02 CC CE	Head of this DATA Length of this DATA One's complement of COMMAND 02 ⊕ CC

Apéndice 6 Lista de comandos del módulo HC-05

AT command Default:

How to set the mode to server (master):
1. Connect PIO11 to high level.
2. Power on, module into command state.
3. Using baud rate 38400, sent the "AT+ROLE=1\r\n" to module, with "OK\r\n" means setting successes.
4. Connect the PIO11 to low level, repower the module, the module work as server (master).

AT commands: (all end with \r\n)

1. Test command:

Command	Respond	Parameter
AT	OK	-

2. Reset

Command	Respond	Parameter
AT+RESET	OK	-

3. Get firmware version

Command	Respond	Parameter
AT+VERSION?	+VERSION:<Param> OK	Param : firmware version

Example:
AT+VERSION?\r\n
+VERSION:2.0-20100601
OK

4. Restore default

Command	Respond	Parameter
AT+ORGL	OK	-

Default state:

Slave mode, pin code :1234, device name: H-C-2010-06-01 ,Baud 38400bits/s.

5. Get module address

Command	Respond	Parameter
AT+ADDR?	+ADDR:<Param> OK	Param: address of Bluetooth module

Bluetooth address: NAP: UAP : LAP

Example:

AT+ADDR?\r\n

+ADDR:1234:56:abcdef

OK

6. Set/Check module name:

Command	Respond	Parameter
AT+NAME=<Param>	OK	Param: Bluetooth module name (Default :HC-05)
AT+NAME?	+NAME:<Param> OK (/FAIL)	

Example:

AT+NAME=HC-05\r\n set the module name to "HC-05"

OK

AT+NAME=ITeadStudio\r\n

OK

AT+NAME?\r\n

+NAME: ITeadStudio

OK

7. Get the Bluetooth device name:

Command	Respond	Parameter
AT+RNAME?<Param1>	1. +NAME:<Param2> OK 2. FAIL	Param1,Param 2 : the address of Bluetooth device

Example: (Device address 00:02:72:od:22:24, name: ITead)

AT+RNAME? 0002, 72, od2224\r\n

+RNAME:ITead

OK

8. Set/Check module mode:

Command	Respond	Parameter
AT+ROLE=<Param>	OK	Param:
AT+ ROLE?	+ROLE:<Param>	0- Slave
	OK	1-Master 2-Slave-Loop

9. Set/Check device class

Command	Respond	Parameter
AT+CLASS=<Param>	OK	Param: Device Class
AT+ CLASS?	1. +CLASS:<Param> OK 2. FAIL	

10. Set/Check GIAC (General Inquire Access Code)

Command	Respond	Parameter
AT+IAC=<Param>	1.OK 2. FAIL	Param: GIAC (Default : 9e8b33)
AT+IAC	+IAC:<Param> OK	

Example:

AT+IAC=9e8b3f\r\n

OK

AT+IAC?\r\n

+IAC: 9e8b3f

OK

11. Set/Check -- Query access patterns

Command	Respond	Parameter
AT+INQM=<Param>,<Param2>,<Param3>	1.OK 2. FAIL	Param: 0——inquiry_mode_standard 1——inquiry_mode_rssi Param2: Maximum number of Bluetooth devices to respond to Param3: Timeout (1-48 : 1.28s to 61.44s)
AT+ INQM?	+INQM : <Param>,<Param2>,<Param3> OK	

Example:

AT+INQM=1,9,48\r\n

OK

AT+INQM\r\n

+INQM:1, 9, 48

OK

12. Set/Check PIN code:

Command	Respond	Parameter
AT+PSWD=<Param>	OK	Param: PIN code (Default 1234)
AT+ PSWD?	+ PSWD : <Param> OK	

13. Set/Check serial parameter:

Command	Respond	Parameter
AT+UART=<Param>,<Param2>,<Param3>	OK	Param1: Baud Param2: Stop bit Param3: Parity
AT+ UART?	+UART=<Param>,<Param2>,<Param3> OK	

Example:

AT+UART=115200，1,2,\r\n

OK

AT+UART?

+UART:115200,1,2

OK

14. Set/Check connect mode:

Command	Respond	Parameter
AT+CMODE=<Param>	OK	Param: 0 - connect fixed address 1 - connect any address 2 - slave-Loop
AT+ CMODE?	+ CMODE:<Param> OK	

15. Set/Check fixed address:

Command	Respond	Parameter
AT+BIND=<Param>	OK	Param: Fixed address (Default 00:00:00:00:00:00)
AT+ BIND?	+ BIND:<Param> OK	

Example:

AT+BIND=1234，56，abcdef\r\n

OK

AT+BIND?\r\n

+BIND:1234:56:abcdef

OK

16. Set/Check LED I/O

Command	Respond	Parameter
AT+POLAR=<Param1,<Param2>	OK	Param1: 0- PIO8 low drive LED 1- PIO8 high drive LED
AT+ POLAR?	+ POLAR=<Param1>,<Param2> OK	
		Param2: 0- PIO9 low drive LED 1- PIO9 high drive LED

17. Set PIO output

Command	Respond	Parameter
AT+PIO=<Param1>,<Param2>	OK	Param1: PIO number Param2: PIO level 0- low 1- high

Example:

1. PIO10 output high level

AT+PI0=10，1\r\n

OK

18. Set/Check – scan parameter

Command	Respond	Parameter
AT+IPSCAN=<Param1>,<Param2>,<Param3>,<Param4>	OK	Param1: Query time interval Param2：Query duration Param3：Paging interval Param4：Call duration
AT+IPSCAN?	+IPSCAN:<Param1>,<Param2>,<Param3>,<Param4> OK	

Example:

AT+IPSCAN =1234,500,1200,250\r\n

OK

AT+IPSCAN?

+IPSCAN:1234,500,1200,250

19. Set/Check – SHIFF parameter

Command	Respond	Parameter
AT+SNIFF=<Param1>,<Param2>,<Param3>,<Param4>	OK	Param1: Max time Param2: Min time Param3: Retry time Param4: Time out
AT+ SNIFF?	+SNIFF:<Param1>,<Param2>,<Param3>,<Param4> OK	

20. Set/Check security mode

Command	Respond	Parameter
AT+SENM=<Param1>,<Param2>	1. OK 2. FAIL	Param1: 0——sec_mode0+off 1——sec_mode1+non_se
AT+ SENM?	+ SENM:<Param1>,<Param2>	

	OK	cure 2——sec_mode2_service 3——sec_mode3_link 4——sec_mode_unknow n Param2: 0——hci_enc_mode_off 1——hci_enc_mode_pt_t o_pt 2——hci_enc_mode_pt_t o_pt_and_bcast

21. Delete Authenticated Device

Command	Respond	Parameter
AT+PMSAD=<Param>	OK	Param: Authenticated Device Address

Example:

AT+PMSAD =1234,56,abcdef\r\n

OK

22. Delete All Authenticated Device

Command	Respond	Parameter
AT+ RMAAD	OK	-

23. Search Authenticated Device

Command	Respond	Parameter
AT+FSAD=<Param>	1. OK 2. FAIL	Param: Device address

24. Get Authenticated Device Count

Command	Respond	Parameter
AT+ADCN?	+ADCN: <Param> OK	Param: Device Count

25. Most Recently Used Authenticated Device

Command	Respond	Parameter
AT+MRAD?	+ MRAD: <Param> OK	Param: Recently Authenticated Device Address

26. Get the module working state

Command	Respond	Parameter

AT+ STATE?	+ STATE: <Param> OK	Param: "INITIALIZED" "READY" "PAIRABLE" "PAIRED" "INQUIRING" "CONNECTING" "CONNECTED" "DISCONNECTED" "NUKNOW"

27. Initialize the SPP profile lib

Command	Respond	Parameter
AT+INIT	1. OK 2. FAIL	-

28. Inquiry Bluetooth Device

Command	Respond	Parameter
AT+INQ	+INQ: <Param1>, <Param2>, <Param3> .... OK	Param1: Address Param2: Device Class Param3 : RSSI Signal strength

Example:

AT+INIT\r\n

OK

AT+IAC=9e8b33\r\n

OK

AT+CLASS=0\r\n

AT+INQM=1,9,48\r\n

At+INQ\r\n

+INQ:2:72:D2224,3E0104,FFBC

+INQ:1234:56:0,1F1F,FFC1

+INQ:1234:56:0,1F1F,FFC0

+INQ:1234:56:0,1F1F,FFC1

+INQ:2:72:D2224,3F0104,FFAD

+INQ:1234:56:0,1F1F,FFBE

+INQ:1234:56:0,1F1F,FFC2

+INQ:1234:56:0,1F1F,FFBE

+INQ:2:72:D2224,3F0104,FFBC

OK

28. Cancel Inquiring Bluetooth Device

Command	Respond	Parameter
AT+ INQC	OK	-

29. Equipment Matching

Command	Respond	Parameter
AT+PAIR=<Param1>,<Param2>	1. OK 2. FAIL	Param1: Device Address Param2: Time out

30. Connect Device

Command	Respond	Parameter
AT+LINK=<Param>	1. OK 2. FAIL	Param: Device Address

Example:

AT+FSAD=1234,56,abcdef\r\n

OK

AT+LINK=1234,56,abcdef\r\n

OK

31. Disconnect

Command	Respond	Parameter
AT+DISC	1. +DISC:SUCCESS OK 2. +DISC:LINK_LOSS OK 3. +DISC:NO_SLC OK 4. +DISC:TIMEOUT OK 5. +DISC:ERROR OK	Param: Device Address

32. Energy-saving mode

Command	Respond	Parameter
AT+ENSNIFF=<Param>	OK	Param: Device Address

33. Exerts Energy-saving mode

Command	Respond	Parameter
AT+ EXSNIFF =<Param>	OK	Param: Device Address

APENDICE 7 GLOSARIO

EEPROM.- (Electrically Erasable Progammable Read Only Memory) Memoria de sólo lectura programable y borrable eléctricamente. Chip de memoria que retiene su contenido sin energía. Puede borrarse, tanto dentro del computador como externamente. Por lo general requiere más voltaje para el borrado que el común de +5 voltios usado en circuitos lógicos. Funciona como RAM no volátil, pero grabar en EEPROM es mucho más lento que hacerlo en RAM.

BROADCAST .- difusión en español, es una forma de transmisión de información donde un nodo emisor envía información a una multitud de nodos receptores de manera simultánea, sin necesidad de reproducir la misma transmisión nodo por nodo

PIC.- Los PIC son una familia de microcontroladores tipo RISC fabricados por Microchip y derivados del PIC1650, originalmente desarrollado por la división de microelectrónica de General Instrument.
El nombre actual no es un acrónimo. En realidad, el nombre completo esPICmicro, aunque generalmente se utiliza como Peripheral InterfaceController (controlador de interfaz periférico).

Piconet.-

Una red de dispositivos que se conectan utilizando Bluetooth. Una piconet puede constar de dos a ocho dispositivos. En una piconet, habrá siempre un master y los demás serán esclavos

TAG.- (etiqueta, mandato, marca) Es una etiqueta o marca que son básicas en lo que se refiere a la programación HTML de las páginas de Internet. Cada uno de estos comandos es interpretado por el navegador para que la información se despliegue de manera adecuada en la pantalla

TRANSPONDEDOR O TRANSPONDER.- es un tipo de dispositivo utilizado en telecomunicaciones cuyo nombre viene de la fusión de las palabras inglesas *Transmitter* (Transmisor) y *Responder* (Contestador/Respondedor)

Apéndice 8 Imágenes de los Códigos QR Utilizados

01….1.

01…5

01…9

02…2

02…6

02…10

03…3

03…7

03…11

04…4

04…8

04…12

Apendice 9 Código MATLAB

Función principal

```
function varargout = InterfazKUKA1(varargin)
% INTERFAZKUKA1 MATLAB code for InterfazKUKA1.fig
%      INTERFAZKUKA1, by itself, creates a new INTERFAZKUKA1 or raises
the existing
%      singleton*.
%
%      H = INTERFAZKUKA1 returns the handle to a new INTERFAZKUKA1 or the
handle to
%      the existing singleton*.
%
%      INTERFAZKUKA1('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in INTERFAZKUKA1.M with the given input
arguments.
%
%      INTERFAZKUKA1('Property','Value',...) creates a new INTERFAZKUKA1
or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before InterfazKUKA1_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to InterfazKUKA1_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 InterfazKUKA1


% Last Modified by GUIDE v2.5 01-Aug-2013 14:29:01

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @InterfazKUKA1_OpeningFcn, ...
                   'gui_OutputFcn',  @InterfazKUKA1_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 InterfazKUKA1 is made visible.
function InterfazKUKA1_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for InterfazKUKA1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% VARIABLES GLOBALES
  assignin('base','UsarImagen',0); %crear la variable de control para
acomodar piezas
  assignin('base','Detener',0);  % variable para la funcion del boton
aceptar
%   assignin('base','Modo',1);
%Clolocar el logo de KUKA en un axes llamado logo2
set(handles.logo2,'visible','off');
set(handles.mapa,'visible','off');
set(handles.aceptar,'visible','off');
set(handles.pushbutton2,'visible','off');
set(handles.text1,'visible','off');
set(handles.checkbox2,'visible','off');
set(handles.checkbox3,'visible','off');
set(handles.IniciarAjuste,'visible','off');
set(handles.Cancelar,'visible','off');

set(handles.checkbox1,'visible','off');
set(handles.text1,'String','COMENZAR?');

axes(handles.LogoCIO)
handles.imagen=imread('logocio.jpg');
imagesc(handles.imagen)
axis off

axes(handles.logo)
handles.imagen=imread('KUKA-Logo-p50.jpg');
imagesc(handles.imagen)
axis off




 assignin('base','check',0); %crear la variable de control para la
seleccion DE simular/ejecutar
set(handles.checkbox2,'value',1); % Por default, opcion simular (Por
seguridad).
% UIWAIT makes InterfazKUKA1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
clc;
clear all;



% --- Outputs from this function are returned to the command line.
function varargout = InterfazKUKA1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
UsarImagen=evalin('base','UsarImagen');%Leer variable global
UsarImagen=UsarImagen+1;
if UsarImagen>1
    UsarImagen=0;
end
assignin('base','UsarImagen',UsarImagen); % Actualizar variable  lobal
% Hint: get(hObject,'Value') returns toggle state of checkbox1



% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
warning off;
set(handles.pushbutton2,'visible','on');
set(handles.aceptar,'Enable','off');
set(handles.Ajuste,'Enable','off');% Deshabilitar menu de Ajuste

UsarImagen=evalin('base','UsarImagen');%Leer variable global
video_obj=evalin('base','video_obj');

 set(handles.Modo,'Enable','off');% Deshabilitar menu durante la lectura
de datos
 set(handles.checkbox1,'Enable','off');% Deshabilitar menu durante la
lectura de datos
 set(handles.salir,'Enable','off');    %Deshabilitar sair bruscamente
 set(handles.checkbox2,'Enable','off');% Deshabilitar menu durante la
lectura de datos
 set(handles.checkbox3,'Enable','off');% Deshabilitar menu durante la
lectura de datos

% tomar imagen de lugares disponibles
[areas,imagen]=Tomar_foto(0,video_obj);
set(handles.mapa,'visible','on');
axes(handles.mapa)
```

```
handles.imagen=imagen;  %imread('matriz.jpg');
imagesc(handles.imagen)
axis off
assignin('base','Detener',0);
Detener=evalin('base','Detener'); %Leer variable global
Modo=evalin('base','Modo');       %Leer variable global



if Detener==0
set(handles.pushbutton2,'Enable','on');
 set(handles.text1,'String','INICIANDO...');
 pause(0.5);

   if Modo==1 %  ARRANCAR BLUETOOTH
              B = Bluetooth('HC-05',1,'Timeout',2);     %configuracion
del objeto. TimeOut(Segundos)
                fopen(B);
                pause(0.5);
              set(handles.text1,'String','Buscando..');
pause(0.5);

         disp('Listo para leer');
   else
          % ARRANCAR CAMARA
          fmt_cam=imaqhwinfo('winvideo',1);
%antes 2
%        video_2 = videoinput('winvideo',2,'YUY2_640x480');
%        video_2 = videoinput('winvideo',1,'MJPG_160x120');
video_2 = videoinput('winvideo',1,'RGB24_320x240');
%antes 2
         % imaqhwinfo
%         preview(video_2);
          start(video_2);
           assignin('base','video_2',video_2);
        javaaddpath('C:\Users\CIO\Desktop\pEquipo1_LECTOR MENSAJES
QR_KUKA\zxing-2.1\core\core.jar');
        javaaddpath('C:\Users\CIO\Desktop\pEquipo1_LECTOR MENSAJES
QR_KUKA\zxing-2.1\javase\javase.jar');
        javaaddpath('C:\Users\CIO\Desktop\pEquipo1_LECTOR MENSAJES
QR_KUKA\zxing-2.1\core\src');
       set(handles.text1,'String','Buscando..');

end
end


%

while Detener==0
%% ADQUISICION DE DATOS
if Modo==1
    [tipo,Lugar]=bluetooth(B) % Adquirir los datos del bluetooth
else
```

```
    [tipo,Lugar]=CodigoQR_Beta(video_2) % Adquirir los datos de la imagen
de QR
end
 %% COLOCAR PIEZAS
if tipo~='0'    % Si un elemento es identificado..
 video_obj=evalin('base','video_obj');
 set(handles.texto_simulacion,'String','');%% Repotar al usuario En caso
de simulacion-------------------OJO ADECUAR

 % TOMAR FOTO DE LUGARES DISPONIBLES
[matriz,imagen]=Tomar_foto(0,video_obj);
% CARGAR LA FOTO AL MAPA
axes(handles.mapa)
handles.imagen=imagen;  %imread('matriz.jpg');
imagesc(handles.imagen);
colormap('gray');
axis off
 check=evalin('base','check');


tipo=str2num(tipo);
 switch tipo
    case 1
        Objeto='Rectangulo';
    case 2
        Objeto='Cubo';
    case 3
       Objeto='Circulo';
case 4
       Objeto='Triangulo';
 end
       disp(Objeto);
   set(handles.text1,'String',Objeto);

if UsarImagen==1
        [disponible,leyenda]=acomodar(matriz,tipo,check);
         set(handles.texto_simulacion,'String',leyenda);%% Repotar al
usuario
         if disponible==0;
         set(handles.text1,'String','No Hay Lugar');%% Repotar al usuario
         end
    else
     [disponible,leyenda]=acomodar3(tipo,Lugar,matriz,check);
     set(handles.texto_simulacion,'String',leyenda);%% Repotar al usuario
         if disponible==0;
         set(handles.text1,'String','No Hay Lugar');%% Repotar al usuario
         end
    end

       % TOMAR FOTO DE LUGARES DISPONIBLES
       pause(5); %% deja tiempo para que KUKA salga de la escena
                  [matriz,imagen]=Tomar_foto(0,video_obj);
                  % CARGAR LA FOTO AL MAPA
axes(handles.mapa)
handles.imagen=imagen;  %imread('matriz.jpg');
                  imagesc(handles.imagen);
```

```
                    colormap('gray');
                    axis off

end
set(handles.text1,'String','Buscando..');%% Refrescar la leyenda al
usuario
    pause(0.5);
  Detener=evalin('base','Detener');%Leer variable global
end %termina while

    numeroKUKA(0,0);

  if Modo==1    %% DETENER BLUETOOTH
       fclose(B);
       clear('B');
      set(hObject,'String','Aceptar');

else
      % Detener Camara
      disp('detener camara QR');
      video_2=evalin('base','video_2');
       stop(video_2); clear video_2; ;       % Detener la camara 2

end

assignin('base','Detener',Detener);
 set(handles.text1,'String','DETENIDO');
 set(handles.checkbox1,'Enable','on');% Habilitar menu al detener proceso
set(handles.Modo,'Enable','on');     % Habilitar checkbox al detener
proceso
set(handles.aceptar,'Enable','on');
set(handles.checkbox2,'Enable','on');% Habilitar menu al detener el
proceso
 set(handles.checkbox3,'Enable','on');% Habilitar menu al detener el
proceso
set(handles.salir,'Enable','on');
set(handles.texto_simulacion,'String','');%% Repotar al usuario En caso
de simulacion-------------------OJO ADECUAR
% --------------------------------------------------------------------
function Modo_Callback(hObject, eventdata, handles)
% hObject    handle to Modo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.IniciarAjuste,'visible','off'); % hacer invisible el boton de
inicio de ajuste
%% Cargar la camara
warning off;
imaqhwinfo;% da informacion
imaqhwinfo('winvideo');
fmt_cam=imaqhwinfo('winvideo',2);
%antes 1
try
    video_obj=videoinput('winvideo',2,'YUY2_640x480');
%antes 1
catch
```

```
    video_obj=videoinput('winvideo',2,'MJPG_640x480');
%antes 1
      %video_obj=videoinput('winvideo',1,'YUY2_640x480');
end
start(video_obj);
assignin('base','video_obj',video_obj);



% --------------------------------------------------------------------
function RFID_Callback(hObject, eventdata, handles)
set(handles.Ajuste,'Enable','on');% habilitar menu al comenzar a trabajar
set(handles.logo2,'visible','on');
set(handles.checkbox1,'visible','on');
%
axes(handles.logo2)
handles.imagen=imread('Tag´s.jpg');

imagesc(handles.imagen);
colormap('gray');
axis off
 assignin('base','Modo',1);%Definir el tipo de lecturas
set(handles.text1,'visible','on');
set(handles.text1,'String','COMENZAR?');

set(handles.aceptar,'visible','on');

set(handles.checkbox2,'visible','on');
set(handles.checkbox3,'visible','on');
% --------------------------------------------------------------------
function QR_Callback(hObject, eventdata, handles)
set(handles.logo2,'visible','on');
set(handles.checkbox1,'visible','on');
set(handles.Ajuste,'Enable','on');% habilitar menu al comenzar a trabajar

axes(handles.logo2)
handles.imagen=imread('qr.jpg');
imagesc(handles.imagen);
colormap('gray');
axis off
 assignin('base','Modo',0);  %definir opcion de lecturas
 set(handles.aceptar,'visible','on');

 set(handles.text1,'visible','on');
set(handles.text1,'String','COMENZAR?');

set(handles.checkbox2,'visible','on');
set(handles.checkbox3,'visible','on');



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)

%----------------------DETENER-----------------------------

% handles    structure with handles and user data (see GUIDATA)
set(handles.Ajuste,'Enable','on');% volver a habilitar
set(handles.pushbutton2,'Enable','off');
numeroKUKA(0,0);    %Descanso a KUKA

video_obj=evalin('base','video_obj');
stop(video_obj); clear video_obj;     % Detener la camara 1


 assignin('base','Detener',1);  % variable para la funcion del boton
aceptar
warning 'on';



% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in salir.
function salir_Callback(hObject, eventdata, handles)
% hObject    handle to salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
warning on;
```

```
close all;
clear all;
% delete(handles.InterfazKUKA1) ;


% --- Executes on button press in checkbox2.  ////////////////SIMULAR
function checkbox2_Callback(hObject, eventdata, handles)

set(handles.checkbox3,'value',0);
 check=evalin('base','check');
 check=check+1;
 if check >1
     check=0;
 end
 if check==1
     set(handles.checkbox3,'value',1);
else
     set(handles.checkbox3,'value',0);
 end
   assignin('base','check',check);

% --- Executes on button press in checkbox3.////////////////EJECUTAR
function checkbox3_Callback(hObject, eventdata, handles)


set(handles.checkbox2,'value',0);
 check=evalin('base','check');
 check=check+1;
 if check >1
     check=0;
 end

  if check==1
     set(handles.checkbox2,'value',0);
else
     set(handles.checkbox2,'value',1);
 end
   assignin('base','check',check);


% --------------------------------------------------------------------
function Ajuste_Callback(hObject, eventdata, handles)
%% La funciónservirá para que el usuario ajuste la camara a la mesa de
trabajo.
% Se quitaran todos los objetos. Después el robot colocara piezas en
% posiciones estratégicas. Se activará la cámara 1, y trabajará de forma
% continua para que el usuario mueva la camara.

%% Limpiar pantalla existente
set(handles.logo2,'visible','off');
set(handles.mapa,'visible','off');
set(handles.aceptar,'visible','off');
set(handles.pushbutton2,'visible','off');
set(handles.checkbox2,'visible','off');
set(handles.checkbox3,'visible','off');
```

```
set(handles.checkbox1,'visible','off');
set(handles.text1,'visible','off');
set(handles.Modo,'Enable','off');% inhabilitar menu


ajuste='Para comenzar con el ajuste, es necesario quitar todos los
objetos de la mesa de trabajo';
ajuste2= 'A continuación, colocar los tres cubos en el tapete destinado a
la función';
%mensaje tradicional de usario
 msgbox(ajuste2,'Advertencia','warn');
 msgbox(ajuste,'Advertencia','warn');

 set(handles.IniciarAjuste,'visible','on'); % hacer visible el boton de
inicio de ajuste
 set(handles.Cancelar,'visible','on'); % hacer visible el boton de paro
%  set(handles.Cancelar,'enable','off');  % deshabilitar el boton de paro




% --- Executes on button press in IniciarAjuste.
function IniciarAjuste_Callback(hObject, eventdata, handles)
set(handles.Modo,'Enable','off');% Deshabilitar menu durante el ajuste
set(handles.Ajuste,'Enable','off');% Deshabilitar menu durante el ajuste
set(handles.IniciarAjuste,'enable','off'); % hacer invisible el boton de
inicio de ajuste
% set(handles.Cancelar,'enable','on');  % deshabilitar el boton de paro
  assignin('base','Ajustar',1);  %definir opcion de lecturas
numeroKUKA(15,1);       % Sacar el número 15 con enable =1. (Colocar
Piezas de Ajuste)
pause(5);
numeroKUKA(0,1);       % Sacar el número 0 con enable =1.  Descanso

pause(10);   % Dar tiempo para que el robot alcance a acomodar las piezas
de ajuste.

  %% CARGAR LA CAMARA
warning off;
imaqhwinfo% da informacion
imaqhwinfo('winvideo');
fmt_cam=imaqhwinfo('winvideo',2);
%antes 1
%video_obj=videoinput('winvideo',2,'YUY2_640x480');
   video_obj=videoinput('winvideo',2,'MJPG_640x480');%antes 1
%    try
%        video_obj=videoinput('winvideo',1,'YUY2_640x480');
%       % video_obj=videoinput('winvideo',1,'MJPG_640x480');
%    catch
%        video_obj=videoinput('winvideo',1,'YUY2_640x480');
%    end
start(video_obj);
Ajustar=evalin('base','Ajustar');
```

```
while Ajustar==1 %% Se entretiene tomando fotos mientras ajustas la
camara
[areas,imagen]=Tomar_foto(0,video_obj);

axes(handles.mapa)
handles.imagen=imagen;  %imread('matriz.jpg');
imagesc(handles.imagen);
colormap('gray');
axis off

% imagesc(imagen);
% colormap(gray);
Ajustar=evalin('base','Ajustar');
 pause(0.2);
end


stop(video_obj);  imaqreset;  % Detener la camara
clear video_obj;
disp('cancleando camara de ajuste');



% --- Executes on button press in Cancelar.
%Cancela la captura de imagenes para ajustar la camara.

function Cancelar_Callback(hObject, eventdata, handles)
set(handles.IniciarAjuste,'enable','on'); % hacer invisible el boton de
inicio de ajuste
assignin('base','Ajustar',0);  %definir opcion de lecturas
 set(handles.IniciarAjuste,'visible','off'); % no hacer visible el boton
de inicio de ajuste
 set(handles.Cancelar,'visible','off'); % no hacer visible el boton de
paro

 numeroKUKA(14,1);       % Sacar el número 14 con enable =1. (Quitar
piezas de Ajuste)
pause(5);
numeroKUKA(0,1);       % Sacar el número 0 con enable =1.  Descanso
set(handles.Modo,'Enable','on');% Habilitar menu
set(handles.Ajuste,'Enable','on');% Habilitar menu

% hObject    handle to Cancelar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Funciones auxiliares

```
function [areas,imagen]=Tomar_foto(mostrar,video_obj)----------------(1)
%% capturar cuadro
imagen_YUY2=getsnapshot(video_obj);
X=ycbcr2rgb(imagen_YUY2);

%% CORRECCION DE IMAGEN
load('deltaBy.mat','deltaBy');
 load('deltaBx.mat','deltaBx');

image=rgb2gray(X);
x=480;
y=480;

bw = image;%im2bw(image,0.56);
bw=imresize(bw,[x y]);


  for i=1:x
     for j=1:y
          corregida(j,i)=bw( (j+deltaBy(j,i)),(i+deltaBx(j,i)) );
     end
  end


Umbral_img=graythresh(X); bw=im2bw(image,Umbral_img);
[M,N]=size(bw);
 corregida=imresize(corregida,[M,N]);

if mostrar==1  % Muestra o no la imagen rectificada
 figure;
 imagesc(corregida);
 title('Imagen corregida');
colormap gray;

end
Umbral_img=graythresh(corregida); bw=im2bw(corregida,0.4);
bw=not(bw);
bw= imfill(bw,'holes');

%% Limpiando Objetos menores a 60 pixceles
bw=bwareaopen(bw,1000); %bw= imfill(bw,'holes');
if mostrar==1  % Si mostrar =1, abrir figuras
figure; imagesc(bw); colormap(gray);axis equal;;axis off;
title('Corregida');
end

%% Centroides
   s  = regionprops(bw, 'centroid');
centroids = cat(1, s.Centroid);

%% Regiones  cuadriculadas
[R,C]=size(bw);
```

```
for i=1:round(R/3):R
  bw(i,:)=255;
end

for i=1:round(C/4):C
  bw(:,i)=255;
end
%%   Pintar centroides sobre imagen
[centroi_R,centroi_C]=size(centroids);
for i=1:centroi_R
    a=nearest(centroids(i,1));
     b=nearest(centroids(i,2));

     for x=-5:5
    bw(b+x,a)=0;
     end
     for x=-5:5
    bw(b,a+x)=0;
     end
end
%%  Identificar posiciones
areas=zeros(3,4); % matriz de posiciones ocupadas
Cent=size(centroids);

for i=1:Cent(1)            % determinación de las X
    if centroids(i,1)>(3*C/4)
        Lugar(i,2)=4;
    elseif centroids(i,1)>C/2
        Lugar(i,2)=3;
    elseif centroids(i,1)>(C/4)
        Lugar(i,2)=2;
else
        Lugar(i,2)=1;
    end
end
                % determinación de las Y
for i=1:Cent(1)
    if centroids(i,2)>(2*R/3)
        Lugar(i,1)=3;
    elseif centroids(i,2)>(R/3)
        Lugar(i,1)=2;
    else
        Lugar(i,1)=1;
end
    end
    %% Colocación sobre la matriz de posiciones

for i=1:Cent(1)
     areas(Lugar(i,1),Lugar(i,2))=1;
end
imagen=bw;
areas
warning on;


function [tipo,Lugar]=bluetooth(B)------------------------------(2)
```

```
warning off;
  format('short');
 name='';cont=0;
leer=1;
 while leer==1
    name2=dec2hex(fread(B,1));
% name2=dec2hex(name2);
    name=strcat(name,name2);
   if name2~=0
      cont=cont+1;
       if cont==5
if strcmp(name,'AABB2DEDC')   %si lleva 5 lecturas y resulta ser un error
de codigo...
                disp('--------ERROR DE LECTURA-----------');
name='';
                cont=0;
             end
       end
        if cont>=21      %Si lleva 21 lecturas..

tipo=name(9);
            unidades=name(39);
            decenas=name(38);
            if decenas=='F'
                decenas='0';
            end
            Lugar= horzcat(decenas,unidades);
% Col=name(39);
            name='';
            cont=0;
            leer=0;


      end
  else
          %        stopasync(B);
leer=0;
          disp('No hubo lectura');
tipo='0';
          Lugar='0';
   end
   warning on;
end



function [tipo,Lugar]=CodigoQR_Beta(obj2)-------------------------(3)
% DISTANCIA DE LECTURA: 27 CENTIMETROS
pause(2);
%  obj2 = videoinput('winvideo',2);
    frame = getsnapshot(obj2);
    frame =ycbcr2rgb(frame);
%     imwrite(frame,'imaqr.jpg','jpg');
%     test_encode = imread('imaqr.jpg');
      test_encode = frame;
    message = decode_qr(test_encode)
```

```
   try
       if message(31)~='f'
          Lugar=strcat(message(31),message(32));
       else
           Lugar=message(32);
       end
%     tipo=strcat(message(1),message(2));
    tipo=message(2);

   catch
       Lugar='0';
       tipo='0';
   end

   clear obj;
```

```
function [disponible,leyenda]=acomodar(areas,figura,ejecutar)---------(4)
disponible=1;
    if areas(1,figura)==0
%         disp(Objeto);
        disp('Renglon 1');
       semilla=0;

    elseif areas(2,figura)==0
%         disp(Objeto);
        disp('PRenglon 2');
        semilla=4;

    elseif areas(3,figura)==0
%         disp(Objeto);
        disp('Renglon 3');
        semilla=8;
    else
        disp('No hay lugar disponible');
        disponible=0;
        posicion=0;
        semilla=-figura;
    end

NumeroKUKA=semilla+figura
if NumeroKUKA~=0
   leyenda0= numeroKUKA(13,ejecutar);
if ejecutar==1
        pause(2);
       end
    leyenda1=numeroKUKA(NumeroKUKA,ejecutar);
        if ejecutar==1
         pause(12);
        end
```

```
else
leyenda0=('               ');
    leyenda1=('               ');
end
leyenda2=numeroKUKA(0,ejecutar);
%  disp(leyenda);
 leyenda=vertcat('Comandos:      ','
',leyenda0,leyenda1,leyenda2);
end
```

-

---(5)

```
function [disponible,leyenda]= acomodar3(figura,Lugar,areas,ejecutar)
% figura=str2num(figura);
Lugar=str2num(Lugar);

%% Determinar la posicion en la matriz
Lugar
disp('Acomodar3');
if Lugar >8
    semilla=3;
    seed=Lugar-8;
    elseif Lugar >4
    semilla=2;
    seed=Lugar-4;
    else
    semilla=1;
    seed=Lugar;
end


% Verificar si la posicion esta disponible
if areas(semilla,seed)==0
         disponible=1;
leyenda0=numeroKUKA(13,ejecutar); %coger pieza
                if ejecutar==1
                pause(2);
                end
        leyenda1=numeroKUKA(Lugar,ejecutar);% Llevar pieza
                if ejecutar==1
                pause(12);
                end
 else
     disp('No hay lugar disponible');
```

```
     disponible=0;
         leyenda0=('               ');
         leyenda1=('               ');
end
    leyenda2=numeroKUKA(0,ejecutar); %Si el lugar está ocupado, o ya se
cumplió con la encomienda, enviar descanso

leyenda=vertcat('Comandos:      ','
',leyenda0,leyenda1,leyenda2);
%  disp(Objeto);

end



function [leyenda]=numeroKUKA(numero,enable)----------------------(6)
%LA FUNCION MANDA A KUKA EL NUMERO DESEADO SI "ENABLE"=1
%LA FUNCION MUESTRA COMO STRING LA ORDEN A KUKA SI "ENABLE"=0

%% Crea las leyendas para simulacion
switch numero
    case 0
        leyenda='Descanso       ';
    case 13
        leyenda='Coger Pieza    ';
    case 14
        leyenda='Ajuste D Camara';
    case 15
        leyenda='Regresa Piezas ';
    otherwise
        leyenda=horzcat('Colocar en:  ',num2str(numero));
        if length(leyenda)<14
            leyenda=horzcat(leyenda,' ');
        end
end

%% El siguiente codigo es para mandar el "espejo" del dato binario a KUKA
  argumento=dec2bin(numero,4);
  argumento2='';
for i=4:-1:1
      argumento2=strcat(argumento2,argumento(i));
  end
  argumento3=strcat('00000000',argumento2);

%% Parte final. Ejecuta o muestra, según "enable".
switch enable
      case 0
            leyenda2= strcat('PIC_KUKA(',argumento3,')');
disp(leyenda2);
      case 1
            PIC_KUKA(argumento3);
  end
```

REFERENCIAS

[1] Sistema de visión artificial para el reconocimiento y manipulación de objetos utilizando un brazo robot, tesis de maestría.
http://tesis.pucp.edu.pe/repositorio/bitstream/handle/123456789/68/SOBRADO_EDDIE_VISION_ARTIFICIAL_BRAZO_ROBOT.pdf?sequence=2

[2] Fernando Reyes Cortes, ROBÓTICA, control de robots manipuladores, primera edición MARCOMBO, S.A., 2011.

[3] Aníbal Ollero Baturone, ROBÓTICA,manipuladores y robots móviles, primera edición MARCOMBO, S.A., 2001.

[4] Groover, Weiss , ROBÓTICA INDUSTRIAL tecnología, programación y aplicaciones MCGRAW-HILL 1989.

[5] "Un robot ha operado en París a seis pacientes a corazón abierto" El país [México] 22 de mayo de 1998.
http://elpais.com/diario/1998/05/23/portada/895874404_850215.html

[6] "Otro brazo articulado, dirigido desde Mallorca, interviene en Barcelona a un enfermo de hidrocefalia"El país [México] 23 de mayo de 1998.
http://elpais.com/diario/1998/05/23/sociedad/895874403_850215.html

[7]http://spectrum.ieee.org/robotics/industrial-robots/the-rise-of-the-machines/0

[8] Craig, John J. ROBÓTICA tercera edición, PRENTICE HALL MEXICO, 2006.

[9] Robots industriales manipuladores, Iñigo/Vidal, primera edición, Alfaomega.

[10] http://www.codigodebarras.pe/codigo-de-barras-historia/

[11]J. Apolinar Muñoz Rodríguez, "Image encoding based on a damped triangular function and a discrete sequence", Journal of Modern Optics, Vol.59,p. 1581-1590, (2012).

[12]J. Apolinar Muñoz Rodríguez, ''Image encryption based on phase encoding by means of a fringe pattern and computational algorithms" Revista Méxicana de Física, Vol. 52 No.1 p. 53-63, (2006).

[14] http://es.wikipedia.org/wiki/C%C3%B3digo_QR

[13] http://www.itsc.org.sg/pdf/synthesis08/Three_QR_Code.pdf

[15] Ching-yin Law & Simon So, QR Codes in Education, Journal of Educational Technology Development and Exchange, 3(1), 85-100.

[16] Baird, J.L. (1928). "Improvements in or relating to apparatus for transmitting views or images to a distance". Patent #GB292, 185.

[17] Stockman, H. (1948). Communication by Means of Reflected Power. Proceedings of the Institute of Radio Engineers. October. Pages 1196-1204.

[18] http://www.eecs.harvard.edu/cs199r/readings/rfid-article.pdf

[19] http://www.libera.net/uploads/documents/whitepaper_rfid.pdf

[20] http://www.nxp.com/documents/data_sheet/MF1S503x.pdf

[21] http://www.madrimasd.org/informacionidi/biblioteca/publicacion/Vigilancia-tecnologica/descargar_documentos/fichero.asp?id=VT13_RFID.pdf

[22] http://es.wikipedia.org/wiki/RFID

[23] https://sites.google.com/site/aplicacionesmovilesjuan/bluetooth

[24] https://sites.google.com/site/aplicacionesmovilesjuan/bluetooth

[25] Malik Zaka Ullah "AN ANALYSIS OF THE BLUETOOTH TECHNOLOGY, Features, Challenges, Future and Security" Master Thesis, Blekinge Institute of Technology, SWEDEN June 2009. http://www.bth.se/fou/cuppsats.nsf/all/bf95c4a61bb329d9c12575d600449bdb/$file/final%20thesis%20after%20removing%20comments.pdf

[26] http://webuser.hs-furtwangen.de/~heindl/ebte-08ss-bluetooth-Ingo-Puy-Crespo.pdf

[27] http://www.helpy.com.ar/Bluetooth/Que-es-Bluetooth.htm

[28] http://www.xatakafoto.com/guias/distorsion-de-lente-vs-distorsion-de-la-perspectiva

[29] Sánchez Martín, Arias Pérez, González Aguilera, Gómez Lahoz. Análisis aplicado de métodos de calibración de cámaras para usos fotogramétricos (2004). http://www.cartesia.org/geodoc/topcart2004/conferencias/38.pdf

[30] J. Apolinar Muñoz Rodríguez, Online self-camera orientation based on laser metrology and computer algorithms, Optics Communications 284 (2011) 5601–5612.

[31] PAJARES MARTIN-SANZ, GONZALO, VISIÓN POR COMPUTADOR, RA-MA EDITORIAL,2002.

[32] J Apolinar Muñoz Rodríguez, R. Rodríguez-Vera, A Asundi and G Garnica Campos, Shape detection using light line and Bezierapproximation network,Centro de Investigaciones en Óptica, A. C., León, Gto, 37150 México.

[33] http://es.wikipedia.org/wiki/Curva_de_B%C3%A9zier

[34] Datasheet **YHY502CTG** Mifare®.

[35] Enrique Palacios, Fernando Remiro, Lucas J. López ,Microcontrolador_PIC16F8 Desarrollo de Proyectos, RA-MA Editorial, primera edición, 2004.

[36] HC-05 Datasheet Itead studio. http://www.electronicaestudio.com/docs/istd016A.pdf

[37] http://www.codigos-qr.com

[38] Catálogo KUKA robotics
http://www.kuka-robotics.com/mexico/es/products/industrial_robots/low/kr16_2/start.htm

[39] J Apolinar Muñoz Rodríguez, "Recognition of a light line pattern by Hu moments for 3-D reconstruction of a rotated object", Optics and Laser Technology Vol .37, p. 131-138 (2005).

[40]Diódodro Guerra Rodríguez, Situación Actual y perspectivas de la educación en ingeniería en México, Revista de la Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma de Nuevo León Fragmento de conferencia presentada el 19 de Mayo de 1999.
ingenierias.uanl.mx/5/pdf/5_diodoro_Guerra_Situacion_actual.pdf

[41] http://articulos.empleos.clarin.com/el-problema-de-atraer-y-retener-estudiantes-de-ingenieria/?cat=8

[42] Artículo periodístico, Vanguardia, Nov.2013.
http://www.vanguardia.com.mx/tienelaindustriaundeficitdeingenierosenmexico-1867831.html

[43] Artículo periodístico de la red de universidades de habla hispana y portuguesa UNIVERISA, Marzo 2013.
http://noticias.universia.net.mx/empleo/noticia/2013/03/07/1009544/ingenieria-carrera-poco-popular-mexico.html

[44] MONREAL REVUELTA, FRANCO CHUMILLAS, "Análisis de estrategias dirigidas hacia la motivación De alumnos de educación secundaria hacia las materias de carácter tecnológico y titulaciones universitarias de ingeniería".
http://www.murciencia.com/upload/comunicaciones/motivacion_ingenieria.pdf