ELABORACIÓN DE UNIDAD DE CONTROL CINEMÁTICO PARA AUTÓMATA ANTROPOMÓRFICO XR-3 DE CINCO GRADOS DE LIBERTAD DOTADO DE VISION ARTIFICIAL

Falcón Orión
Fernández Mariely
Tutor: Prof. Wilmer Sanz

Universidad de Carabobo
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
Departamento de Sistemas y Automática ELÉCTRICA

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE ELECTRÓNICA Y COMUNICACIONES

ELABORACIÓN DE UNIDAD DE CONTROL CINEMÁTICO PARA AUTÓMATA ANTROPOMÓRFICO XR-3 DE CINCO GRADOS DE LIBERTAD DOTADO DE VISION ARTIFICIAL

TRABAJO ESPECIAL DE GRADO PRESENTADO ANTE LA ILUSTRE UNIVERSIDAD DE CARABOBO PARA OPTAR AL TÍTULO DE INGENIERO ELECTRICISTA

Autores

Falcón Orión Fernández Mariely

Tutor

Prof. Wilmer Sanz

Naguanagua, Junio de 2014

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE ELECTRÓNICA Y COMUNICACIONES

ELABORACIÓN DE UNIDAD DE CONTROL CINEMÁTICO PARA AUTÓMATA ANTROPOMÓRFICO XR-3 DE CINCO GRADOS DE LIBERTAD DOTADO DE VISION ARTIFICIAL

TRABAJO ESPECIAL DE GRADO PRESENTADO ANTE LA ILUSTRE UNIVERSIDAD DE CARABOBO PARA OPTAR AL TÍTULO DE INGENIERO ELECTRICISTA

Autores

Falcón Orión Fernández Mariely

Tutor

Prof. Wilmer Sanz

Naguanagua, Junio de 2014

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA

Certificado de Aprobación

Los abajo firmantes miembros del jurado asignado para evaluar el trabajo especial de grado titulado **«ELABORACIÓN DE LA UNIDAD DE CONTROL CINEMÁTICO PARA AUTÓMATA ANTROPOMÓRFICO XR-3 DE CINCO GRADOS DE LIBERTAD DOTADO DE VISIÓN ARTIFICIAL»**, realizado por los bachilleres: **Falcón V. Orion L. C.I. N° 21215656** , **y Fernández R. Mariely C. C.I. N° 20181083** hacemos constar que hemos revisado y aprobado dicho trabajo.

Jurado: Ing. Teddy Rojas

Jurado: Ing. Liliana Villavicencio

Tutor: Ing. Wilmer Sanz

Naguanagua, Marzo de 2015

Dedicatoria

Este proyecto final de la carrera es un aporte al desarrollo del conocimiento en mi país, producto de mi educación en la Universidad de Carabobo y mi formación como ser humano, por ello, debo agradecer primeramente a mis padres Nelson Falcón y Liliana Valiente sin los cuales no sería quien soy ahora, así también tanto a mis hermanos Antares y Altaír, como a mis mejores amigos y profesores que a momento oportuno contribuyeron a consolidar el camino de mi formación hasta el término de la carrera. Considero que esta obra es una aproximación a un sueño, dedicado a los compañeros entusiastas por la innovación, la robótica o el deseo de "crear"; siendo la presente solo una muestra que desde un rincón del mundo, se pueden hacer cosas geniales.

Orión Falcón

Dedicatoria

Dedico este trabajo a a Dios, a mis padres y a mi familia por haberme apoyado y guiado durante todo el camino en esta carrera. A mis amigos y profesores que estuvieron ahí siempre para brindarme su cariño sincero y apoyo incondicional.

Mariely Fernández

Agradecimientos

Agradecemos a nuestros padres por su apoyo incondicional, a la Universidad de Carabobo y a los profesores que han formado parte de nuestra formación profesional. Hacemos un agradecimiento especial a nuestro tutor Wilmer Sanz, por su confianza, ayuda y orientación a lo largo de la realización de este trabajo.

Orión y Mariely

Índice general

Índice de figuras

Índice de cuadros

Introducción

La vida moderna requiere el uso diario de gran cantidad de artículos y servicios, los cuales deben ser producidos masivamente para atender a una gran población en constante aumento. Para ello la industria del siglo XX empleó herramientas y máquinas de trabajo continuo capaces de lograr mayor precisión, fuerza y eficiencia de tiempo que un ser humano en cada jornada de trabajo. Con la llegada de los sensores de la era digital, marcada por los computadores, las máquinas pudieron interactuar con su entorno y responder en forma automática ante diferentes variables, marcando un paso importante en la evolución industrial. Uno de los más notables automatismos es aquel que por su versatilidad reproduce la movilidad del brazo humano y las herramientas que este pueda emplear, constituido por una cadena controlada de eslabones semejante huesos de un brazo humano; pero con variada movilidad y geometría, es lo que llamamos hoy día "Robots Industriales", usados para todo tipo de tareas en la industria como soldadura, corte, tamizado, pintura, colocación, clasificación y traslado de paquetes entre otros.

En el siglo XXI la robótica se ha convertido en un área estratégica para la automatización industrial, siendo la industria fundamental en el estilo de vida de la sociedad moderna. Con el objetivo de hacer evolucionar la tecnología y formar a profesionales que puedan liderarla en el futuro, abre el camino la Universidad de Carabobo con la línea de investigación: "Robótica y Visión Industrial". Cuyo laboratorio correspondiente posee tres robots semejantes a los más comerciales y más comunes en la industria, mostrando los conceptos aplicados de su funcionamiento y uso en las cátedras de "Robótica y Visión Industrial" y "Control de Procesos por Computadora".

El Rhino XR-3 de Robotics Ltd, es un brazo robot de cinco articulaciones utilizado por educadores que necesitan un diseño abierto, resistente y adecuado para el comienzo de los estudiantes en el área de la Robótica. Un diseño popular con instalaciones en todo el mundo.

El presente trabajo pretende lograr el funcionamiento del robot antropomórfico

modelo XR-3 que se encuentra en el laboratorio de Robótica de la Universidad de Carabobo, implementando una nueva unidad y sistema de control completo, que ademas incluya Visión Artificial, y que permita a los estudiantes pertenecientes a la cátedra aprovechar al máximo el beneficio que puede proporcionar controlar y experimentar con un robot manipulador como el XR-3 en la práctica.

En este trabajo se describe resumidamente como se hizo el diseño y la construcción de la unidad de control cinemático (UCC) y software de control del autómata XR-3 perteneciente al Laboratorio de Robótica y Visión Industrial de la Facultad de Ingeniería, y su contenido está organizado en la siguiente estructura:

Capítulo 1: **El Problema**. En este capítulo se describe al lector el contexto e importancia de la investigación, delimitando las soluciones y objetivos de la propuesta.

Capítulo 2: Marco conceptual. En este capítulo se exponen los basamentos teóricos que permiten abordar la investigación; aportes de los antecedentes más significativos, y exposición de la teoría fundamental para el entendimiento del proyecto.

Capítulo 3: Metodología. Este capítulo ilustra los procedimientos empleados para la ejecución y seguimiento marcando pauta para mejoras futuras o proyectos similares.

Capítulo 4: Diseño del hardware programable. Expone la descripción del circuito que constituye la Unidad de Control Cinemático (UCC) del robot XR-3, así como las consideraciones más importantes de diseño y experiencias durante su desarrollo. El circuito diseñado tiene por objetivos controlar cada uno de los motores para lograr una posición angular deseada, y comunicar el circuito por USB con un computador que le emita órdenes de posición específicos.

Capítulo 5: Diseño del software de control. Muestra el desarrollo, aplicación y consideraciones del modelo matemático planteado para la Cinemática Inversa y Directa del brazo antropomórfico. Además describe a detalle los algoritmos utilizados para hacer el procesamiento de la imagen y poder discriminar objetos de color determinado, junto con sus respectivos resultados. Explica el funcionamiento general de las principales funciones diseñadas para el control del Rhino XR-3 en el software Matlab, y expone el uso adecuado de la librería HIDAPI para lograr una comunicación USB exitosa entre Matlab y el micro-

controlador. También detalla la estructura de la Interfaz Gráfica diseñada y la utilización del software de simulación Roboworks.

Capítulo 6: Conclusiones y recomendaciones.

1 El problema

1.1. Planteamiento del problema

Actualmente en Venezuela pocas empresas emplean robots en sus procesos; esto se debe fundamentalmente al elevado costo de importación, pero también al requerimiento de personal capacitado en automatismos robóticos que los adecuen a procesos específicos.

Cumpliendo con la necesidad nacional de desarrollo industrial, la Universidad de Carabobo realiza labores de investigación y docencia en "Robótica y Visión Industrial", desde un laboratorio de la Facultad de Ingeniería; incorporando la robótica a la formación integral del ingeniero electricista especializado en automatización, proveyéndole de capacitación técnica para la implementación industrial de esta tecnología, pero también la teoría necesaria que fundamente las bases de la fabricación nacional de robots industriales; que disminuya su elevado costo de importación, ya que el conocimiento teórico antecede toda innovación tecnológica práctica, citando a Sanz "*de este modo una idea sobre lo que "puede ser», se presente como lo que lo que "será"o como lo que "es», dentro de un contexto futurista, y en simpatía con tal idea, tarde o temprano, un pionero la toma como inspiración y descubre el modo de hacerla realidad, allanando un camino, descubriendo una senda"*[1], favoreciendo el desarrollo tecnológico del país, y la humanidad.

Para lograr las metas del mencionado laboratorio, se emplean simuladores virtuales y prototipos de los tres robots más frecuentes en la industria mundial: del tipo SCARA; como el usado en Síragon y Vetelca para el ensamblado electrónico, Antropomórfico; como el usado en Chevrolet y Ford de Venezuela en la pintura de chasis automotriz, y el tipo cilíndrico empleado para empaquetado y envasamiento. Sin embargo actualmente los tres se hallan fuera de servicio por daños en sus unidades de control, comprometiendo la funcionalidad y objetivos del laboratorio en docencia e investigación.

Para diseñar un Robot se debe poseer un compendio de habilidades básicas

en campos como la ingeniería electrónica, eléctrica, computación, mecánica e inteligencia artificial, debido a que los sistemas robóticos requieren una combinación de elementos para hacer efectiva la gestión de tareas. En cuanto a la programación de tareas de un robot, cada fabricante provee una interfaz que permite al usuario indicar la posición o trayectoria que desea que el robot realice, sin embargo, consideraciones de esfuerzo, desgaste y trayectoria óptima, dependen de la formación del ingeniero.

La Universidad de Carabobo se encuentra constantemente formando Ingenieros Electricistas con la capacidad de diseñar y programar unidades de control para robots, y es donde la cátedra de «Robótica y Visión Industrial» juega un papel importante. Dicha cátedra actualmente posee en su laboratorio de Robótica y Visión Industrial un manipulador robótico modelo XR-3 fabricado por Rhino Robots.Inc, que se encuentra fuera de funcionamiento.

1.2. Justificación de la investigación

Actualmente la asignatura de Robótica y Visión Industrial dictada en la Escuela de Ingeniería Eléctrica posee en el laboratorio un robot Rhino XR-3 fuera de operación, debido a esto, se torna complicado visualizar los ejemplos dados en las sesiones teóricas, y se dificulta plasmar con hechos lo que ocurre en la realidad al momento de operar y manipular robots. Por esta razón, resultaría de gran importancia y utilidad que este robot este operativo, sin implicar mayores costos para nuestra casa de estudios. El XR-3 ha tenido dos unidades de control, la del fabricante y un trabajo de grado en 2005 [2], la baja robustez de los diseños e implementación electrónica han dejado al XR-3 fuera de servicio por casi 10 años, siendo pertinente una innovación que incorpore nuevas tecnologías electrónicas.

Lograr la funcionalidad del robot Rhino XR-3 implica elaborar una unidad de control que le permita ser operado en cinemática abierta siendo útil en las prácticas de Laboratorio de Robótica y Visión Industrial. Además, se propone añadir al robot un sistema de control mediante un sistema de Visión Artificial por computador y con comunicación vía USB, para facilitar así el uso de este brazo robótico por parte de los estudiantes de la cátedra de Robótica y Visión Industrial.

La elaboración de la mencionada unidad de control pondrá en operación al

robot XR-3, lo que potenciará docencia de la robótica por ser un prototipo para el ensayo de experimentos y pruebas que validen los conocimientos teóricos impartidos en aula, propiciando además la vinculación de la robótica con los procesos industriales de actualidad. La incorporación del procesamiento de imágenes que constituye la visión artificial de la unidad de control propuesta, provee al operador la gestión automática de tareas, convirtiendo al XR-3 en una máquina capaz de reconocer ciertas características. Siendo un avance en la línea de investigación en "Robótica y Visión Industrial" y abriendo camino al desarrollo tecnológico del país.

1.3. Objetivos

1.3.1. Objetivo general

Construir el circuito impreso de la Unidad de Control cinemático para el autómata antropomórfico XR-3 de cinco grados de libertad, el cual pueda ser manipulado mediante interfaz gráfica que le permita trasladar objetos de terminado color y/o forma, empleando visión por computador a través de cámara web.

1.3.2. Objetivos específicos

1. Diseñar el circuito electrónico utilizando un micro-controlador, que permita el control de los motores de cada articulación y garantice el correcto posicionamiento angular de las mismas.
2. Construir el circuito impreso que posea el sistema de control de motores de las articulaciones y se adapte al sistema de comunicación estándar USB.
3. Desarrollar un modelo para la determinación de la cinemática directa e inversa del robot Rhino XR-3.
4. Evaluar el modelo cinemático inverso mediante un software especializado.
5. Discriminar objetos mediante procesamiento de imagen captada por cámara web.

6. Desarrollar una interfaz que permita gestionar tareas al XR-3 a través de visión por computador.

1.4. Alcance y limitaciones de la investigación

El presente trabajo especial de grado se limita a construir una unidad de control para el brazo robot Rhino XR-3, diseñar un sistema de control que incluya Visión Artificial, y la elaboración de una Interfaz Gráfica que permita la comunicación del computador con el robot vía USB.

El sistema de control posee 3 formas de posicionar el robot en un punto. La primera es de forma manual, donde el usuario le debe proporcionar: las coordenadas espaciales de un punto (dando clicks sobre la imagen o simplemente definiéndolas manualmente), la orientación de la pinza, y el estado del pinza (cerrado o abierto), de todos los puntos que pertenezcan a la trayectoria que se desea que siga el brazo robot. La segunda forma consiste en recibir por el usuario los valores de los ángulos de cada articulación y el estado de la pinza (cerrado o abierto). Y en la tercera y ultima forma se tiene que definir un punto $P_f(x, y, z)$ adonde debe llegar un objeto, con su orientación. Este objeto se ubica en un punto $P_i(x, y, x)$, el cual se halla luego de procesar una imagen tomada por una cámara web. El procesamiento de esta imagen está condicionada a que se identifique un solo objeto a la vez, el cual puede ser de color rojo, verde o azul. En cuanto a la forma del mismo podría variar entre cilindros, cubos o paralelepípedos de dimensiones limitadas por la herramienta terminal.

En cuanto al modelo cinemático inverso se determinan los desplazamientos angulares que debe moverse cada articulación para posicionarse en una o varias coordenadas consecutivas, considerando que los puntos en los que el robot posiciona su herramienta terminal, siempre están un plano XY a una determinada Z y tomando en cuenta la restricción del plano de trabajo, que es un área rectangular de 39.4 cm de alto y 53.1 de ancho a la cual el robot puede tener acceso al variar solo 180° de su primera articulación. Ademas debe garantizarse que el sistema de referencia del robot coincida lo más que se pueda con el sistema de referencia del plano de trabajo,

En cuanto a la cámara tipo web para la obtención de la imagen que permite al robot localizar el objeto a transportar, se escogió trabajar con una relación

de aspecto de la imagen es de 3:4, con resolución de 480x640.

El error de posicionamiento en terminos porcentuales no debe ser mayor a $\pm 10\,\%$. La longitud del indicador de posición del objeto de ensayo no debe exceder las dimensiones de 4x4 cm en el plano y 6 cm de alto debido a las limitaciones del actuador utilizado tipo pinza.

No se espera que el robot sea capaz de operar continuamente durante docenas de minutos o manipular cargas superiores a los 300g.

La interfaz es realizada en Matlab 64bits que trabaja sobre plataforma Windows 7 o superior. La simulación completa del funcionamiento del robot se hace utilizando el modelador Roboworks el cual debe recibir un archivo de datos .dat con una estructura específica. Dicho archivo es generado mediante el uso de la Interfaz Gráfica desarrollada en Matlab.

2 Marco Teórico

2.1. Antecedentes

Para fundamentar la investigación, a continuación se señalan los recursos bibliográficos (trabajos de grado, libros impresos, publicaciones de revistas, etc.) que son fuentes de información que sirven como apoyo para el desarrollo del presente trabajo de grado.

Frassato A. y Trujillo D. en su trabajo especial de grado "Controlador de brazo robótico" del año 1995, fundamentaron la base de la investigación bibliográfica, siendo uno de los antecedentes más cercanos a la elaboración de unidades de control para este tipo de robot de cinco grados de libertad, permitiendo mayor seguridad en el manejo del hardware. Sin embargo, la tecnología de componentes empleados en este trabajos se considera anticuada, y se debe recurrir al re-diseño de la unidad de control completamente.

De manera similar, Colina Fausto (2005) autor del trabajo especial de grado "Diseño e implementación de una interfaz controladora para manipulador robótico Rhino XR-3 por comunicación paralela", consistió en colocar en operación al XR-3 tras varios años fuera de servicio. El complicado hardware de control utilizado por los diseñadores anteriores, y la cercanía del transformador de alimentación a la tarjeta de control produjeron el colapso del controlador sacándolo de operación; sus fallas sirven de experiencia para la presente investigación, y así lograr el desarrollo de un hardware más robusto y duradero, teniendo previsión de aislar adecuadamente las etapas lógica y analógica.

En el artículo escrito por Soares L. y Casanova V. (2006) titulado "An educational robotic workstation based on the rhino xr4 robot" se describe una estación de trabajo robótica educativa desarrollada para experimentos de laboratorio sobre la manipulación robótica. Hicieron el diseño y la construcción de un controlador para el Rhino XR-4, así como también elaboraron una toolbox para Matlab que incluye funciones robóticas genéricas y específicas para

el robot. A lo largo de este trabajo los autores describen el control por computador del robot XR-4, y muestran un esquema de multiplexación paralela de los encoder de los motores, además exponen información relevante respecto a los lazos de control requeridos en dicho diseño[3]. La contribución de este artículo va enfocada hacia estos modelos y esquemas antes descritos, que serán tomados como referencia a seguir para la investigación.

El artículo publicado por Cuevas E., Rojas R., y Zaldivar D en el 2003 titulado "Computer vision using Matlab and the toolbox of image processing", junto con el trabajo de Rojas T. "Diseño de un sistema de visión artificial para un manipulador industrial antropomórfico" (2008) , fundamentaron las bases para la implementación del sistema de visión artificial, ya que profundizan acerca de la implementación de la toolbox que posee Matlab para el procesamiento de imágenes. Muestra la forma en que pueden ser utilizados algoritmos propios de Matlab, para manipular imágenes, como por ejemplo conceptos básicos de imágenes, acceso a píxeles en el plano, muestreo y filtrado de imágenes, funciones de detección de bordes, operaciones morfológicas, imágenes binarias y segmentación por umbral, entre otras.

Falcón O., Quevedo E., Reyes J, Sanz W. en su trabajo "Brazo robótico planar de dos grados de libertad" (2013) y Adrián E. junto a Castañeda A. en el trabajo especial de grado "Implementación de un sistema de manipulación y control mediante visión artificial para un autómata industrial SCARA", realizaron aportes que permitieron consolidar la teoría y práctica de la implementación robótica, construcción del hardware y software que requiere una unidad de control para robots de dos y tres grados de libertad. Adicionalmente el trabajo de Adrián y Castañeda es el primer antecedente de implementación de visión por computador, aunque no involucra generación automática de tareas.

En la publicación realizada por KUKA ROBOTER GMBH (2013) titulada "Robot arm with an adjustment device" describen a un brazo robot de la empresa KUKA, el cual tiene la disposición antropomórfica que caracteriza a los brazos robots de este fabricante, siendo este el caso particular más moderno de un robot de disposición de articulaciones antropomórficas dispuestas de la misma forma que el robot XR-3. La información presentada a lo largo de la publicación constituye una referencia importante para la presente investigación, en cuanto a la aplicación del controlador que se requiere desarrollar.

2.2. Robot Manipulador Antropomórfico XR-3

2.2.1. Manipuladores Antropomórficos

La mayor parte de los robots industriales son esencialmente brazos articulados. De hecho según la definición del "Robot Institute of América", un Robot Industrial es un manipulador programable multifuncional diseñado para mover materiales, piezas, herramientas o dispositivos especiales, mediante movimientos variados, programados para la ejecución de distintas tareas. De acuerdo con su grado de autonomía, los robots pueden clasificarse en teleoperados, de funcionamiento repetitivo y autónomo o inteligentes.

En los robots teleoperados las tareas de percepción del entorno, planificación y manipulación compleja son realizadas por humanos. Es decir, el operador actúa en tiempo real cerrando un bucle de control de alto nivel. Los sistemas evolucionados suministran al operador re-alimentación sensorial del entorno (imágenes, fuerzas, distancias). En manipulación se emplean brazos y manos antropomórficas con controladores automáticos que producen los movimientos del operador.

Los robots manipuladores son, esencialmente, brazos articulados. De forma más precisa, un manipulador industrial convencional es una cadena cinemática abierta formada por un conjunto de eslabones o elementos de la cadena interrelacionados mediante articulaciones o pares cinemáticas, tal como se ilustra en la Figura 2.1. Las articulaciones permiten el movimiento relativo entre los sucesivos eslabones.

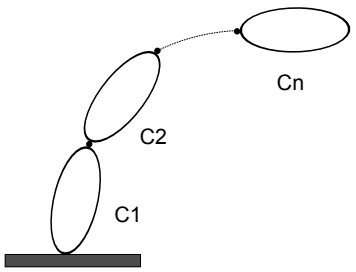


Figura 2.1: Cadena Cinemática abierta
Fuente: A. Ollero (2001)[4]

Existen diferentes tipos de articulaciones, siendo las más utilizadas en robótica las que se indican en la Figura 2.2. La articulación de rotación suministra un

grado de libertad consistente en una rotación alrededor del eje de la articulación. Esta articulación es, con diferencia la más empleada.

En la articulación prismática el grado de libertad consiste en una traslación a lo largo del eje de la articulación. En la articulación cilíndrica existen dos grados de libertad: una rotación y una traslación. La articulación planar está caracterizada por el movimiento de desplazamiento en un plano, existiendo, por tanto, dos grados de libertad. Por último, la articulación esférica combina tres giros en tres direcciones perpendiculares en el espacio [4].

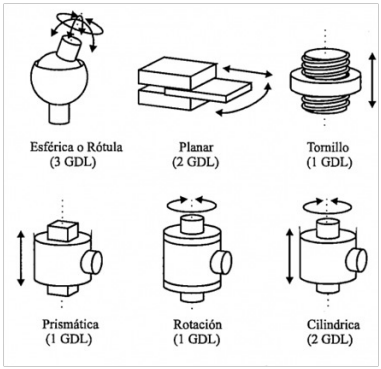


Figura 2.2: Tipos de articulaciones
Fuente: A. Ollero (2001) [4]

2.2.2. Descripción cinemática del robot de 5 grados de libertad

El robot RHINO XR-3 posee cinco articulaciones, todas ellas son de tipo rotoide. La articulación 1 mueve la base del robot hasta 265^o al alrededor del eje Z, las articulaciones dos, tres y cuatro se encargan de posicionar el brazo para alcanzar diferentes puntos en el volumen de trabajo de dicho robot, y la articulación cinco es la que se encarga de controlar la orientación de la herramienta de trabajo.

El efector final de este robot puede alcanzar todos los puntos internos a una esfera de radio igual a la suma de la longitud de las distancias entra las articulaciones 2, 3 y 4.

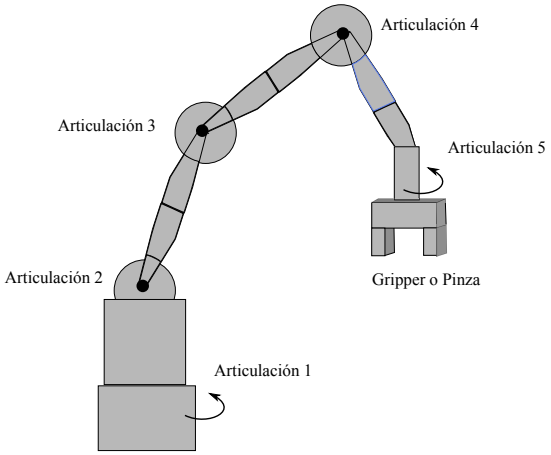


Figura 2.3: Modelo aproximado del Robot Rhino XR-3

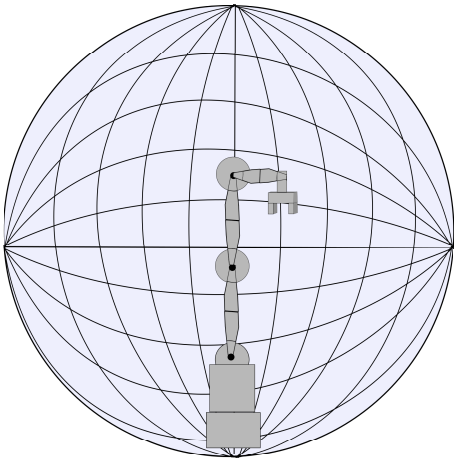


Figura 2.4: Alcance aproximado del robot Rhino XR-3

2.2.3. Robot comercial XR-3.

El robot modelo XR-3 es un sistema robótico desarrollado y fabricado por Rhino Robotics LTD., empresa con sede en Miamitown en USA. El robot manipulador de tareas ligeras fue diseñado para aplicaciones de investigación y docencia en laboratorio, hábil para la manipulación de objetos y herramientas, es útil para la planificación de tareas y trayectorias de robots antropomórficos industriales.

El sistema robótico se encuentra constituido por una estructura mecánica articulada llamada XR-3 Robot Arm, que consta de una pinza y cinco articulaciones, movidas por motores DC con encoder, que le permiten el movimiento geométrico de un robot antropomórfico, abarcando un volumen de trabajo similar a una esfera de 61cm de radio ubicado a 20cm de su base, ver (b) en la Figura 2.5 ,pudiendo levantar un peso máximo de 0,45Kg, con su estructura

de peso neto 7,65Kg [5].

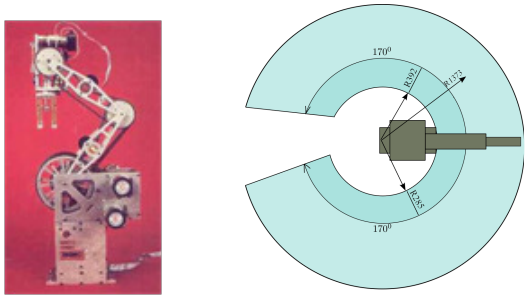


Figura 2.5: (a) Fotografía del robot manipulador antropomórfico XR-3 (b) Área de trabajo
Fuente: Rhino Robotics LTD[5]

2.2.4. Especificaciones Técnicas.

Cada una de las 5 articulaciones y el actuador pinza son movidos por motores DC, para un total de seis (6) motores DC cada uno con dos encoder en cuadratura detrás de cada uno, el eje de los motores tiene una caja de engranajes convertidora de velocidad a par mecánico, estando además acoplados cuatro de estos motores a cadenas engranadas con diferente número de dientes para tal fin, cada motor identificado con una letra de la A a la F, tiene especificaciones teóricas dadas por el del fabricante resumidas en el Cuadro 2.1 y el Cuadro 2.2.

Cuadro 2.1: Relación de engranajes y velocidad

Articulación	Motor	Resolución en Grados	Relación de Engranajes	No. dientes piñón	Vel.	Tipo de Motor
Mano	A	No aplica	96:1	No aplica	1	Small
Rotación de la muñeca	B	0,18	165,4:1	5,51	32	Small
Flexión de la muñeca	C	0,12	66,1:1	8,8	45	Large
Codo	D	0.12	66,1:1	8,8	30	Large
Hombro	E	0.12	66,1:1	8,8	20	Large
Cintura	F	0,23	66,1:1	4,4	60	Large

En el trabajo especial de grado de Frassato & Trujillo se realizó una curva de vacío para los motores del XR-3, la cual se muestra en la Figura 2.6, en la que

Cuadro 2.2: Capacidades de los motores

Motor	Par nom. (N-m)	Par pico (N-m)	Volt. ref. (V)	Corrien. vacio (A)	Corrien. pico (A)	Max par caja red. (N-m)	Vel. vacio caja red. (RPM)
Small	0.0089	0.0357	12-24	0.11	1.40	0.71	86.5
Large	0.0226	0.110	12-20	0.20	5.54	1.24	85.3

se destaca la velocidad de 4250 RPM para una tensión de 18V, sobre el codo de saturación del entrehierro de la máquina hasta donde el comportamiento es aproximadamente lineal, los autores recomiendan mantener una tensión de 18V como alimentador de motores.

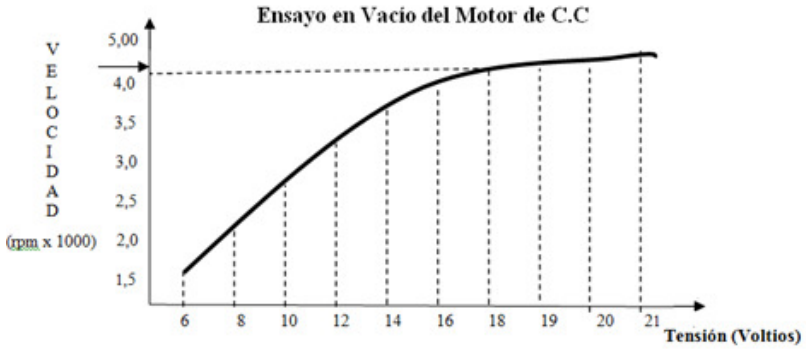


Figura 2.6: Curva de vacío de los motores del XR-3
Fuente: Frassato-Trujillo (1995) [6]

2.2.5. Unidad de control del Rhino XR-3

La unidad de control del robot es el sistema capaz de controlar cada motor para que el robot realice una acción determinada que el usuario especifique. Para el Robot comercial XR-3 el fabricante desarrolló el "Mark III controller", una caja de $80 \times 40 \times 15$ centímetros en la cual se encuentra el circuito que controla las articulaciones del robot como lo muestra la Figura 2.7. Dicha unidad puede ser controlada desde un computador, a través del puerto paralelo, mediante el software "Mark III". Sin embargo también es posible controlar los motores mediante RS232C con un control manual "Teach pendant". En ambos casos el control del robot es mediante cinemática directa, tal como si se tratase de un carro de juguete a radio-control con rutina programable. Con respecto al funcionamiento del "Mark III controller" es propiedad intelectual de Rhino

Robotics. LTD; sin embargo se sabe que funciona como procesos paralelos mediante el sistema “ECE565 Robotics class Versión 2.0”.

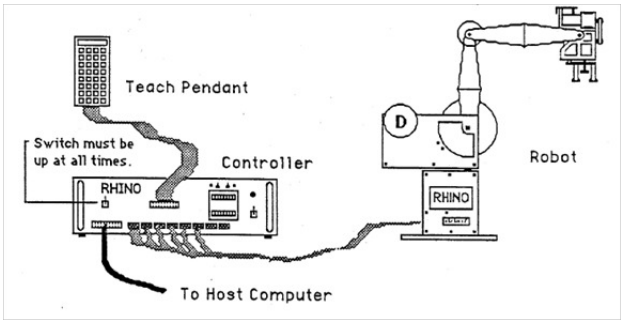
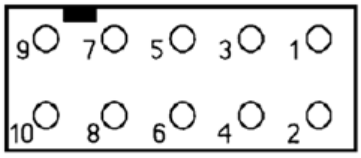


Figura 2.7: Vista completa del robot con su unidad de control
Fuente: Rhino Robotics LTD[5]

2.2.6. Conectores.

La alimentación de los motores esta constituida por un interruptor de calibración (“Limit Switch”) y su encoder diferencial (“Optic A” y “Optic B”), los cuales se encuentran embebidos en cada articulación del robot, y se tiene acceso a los contactos mediante un cable de 10 pines descrito en la Figura 2.8.



1. Logic Ground
2. Optic "B"
3. +5 Volts
4. Optic "A"
5. N/C
6. Limit Switch
7. Motor Common
8. Motor

Figura 2.8: Descripción del cable de alimentación de los motores del Rhino XR-3.
Fuente: Rhino Robotics LTD (1992)[5]

2.3. Control de motores de corriente continua.

El motor de corriente continua (Direct Current abreviado DC) es una máquina que convierte la energía eléctrica en mecánica como un movimiento giratorio. Mediante la circulación de corriente por un bobinado, se genera un campo

magnético que en presencia de imanes permanentes genera una fuerza capaz de hacer rotar el eje de giro de la máquina.

2.3.1. Modelación para un motor DC de baja potencia.

La potencia eléctrica en corriente continua es $P_e = VI$ (voltaje por corriente). La potencia mecánica angular es $P_m = T\omega$ (Par mecánico por velocidad angular). De la ley de Faraday se deduce $V = k_1\omega$, y de la Ley de Ampere $I = k_2T$, donde k_1 y k_2 son variables que dependen de la histéresis del medio ferromagnético, que en primera aproximación pueden considerarse constantes. Logrando de esta manera una relación proporcional entre variables eléctricas y mecánicas.

2.3.2. Medición de velocidad y corriente

Un encoder es un disco marcado a modo de transportador que acoplado al eje de un motor permite contar los ángulos que ha girado el motor usando un sensor óptico. La señal observada al girar el motor es una señal cuadrada cuyo período depende de la velocidad del eje; sin embargo es conveniente emplear dos sensores con marcas en el disco en cuadratura que permitan saber el sentido de giro del eje del motor. El Rhino XR-3 emplea un encoder de cuadratura con salidas “optic A” y “optic B” (ver Figura 2.9). Para medir la corriente puede emplearse una resistencia y el convertidor de voltaje analógico-digital (ADC) de un microcontrolador [7] .

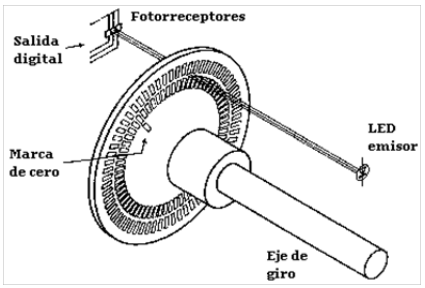


Figura 2.9: Constitución de un encoder
Fuente: Rhino Robotics LTD (1992)[5]

2.3.3. Dominio de motores DC.

El par mecánico realizado por el motor define el sentido de giro y es proporcional a la corriente, el sentido de la corriente es definido por la polarización de tensión que es aplicado en terminales del motor, valiéndose de esta circunstancia se ha desarrollado el arreglo llamado "puente H" mostrado en (a) Figura 2.10; donde se observa que el sentido de la corriente que alimenta al motor puede ser cambiada accionando los pares de interruptores S1 y S4 ó el par S2 y S3. Existen formas de implementar este arreglo como un circuito eléctrico empleando transistores BJT o MOSFET como interruptores electrónicos.

Debido a que la velocidad de giro es proporcional a la tensión aplicada, puede inducirse el frenado aplicando menor tensión, equivalente a menor potencia. Para lograr esto digitalmente, se puede emplear el hecho de que las variables eléctricas pueden variar rápidamente y las mecánicas no. Por lo tanto, si la tensión es una señal rectangular de 1KHz, el motor responderá en realidad al valor promedio de la señal. De este modo, variando la duración del intervalo diferente de cero se puede variar la tensión aplicada al motor, y con ello su velocidad. A esto se denomina modulación de ancho de pulsos o (PWM, pulse width modulation) (ver (b) Figura 2.10).

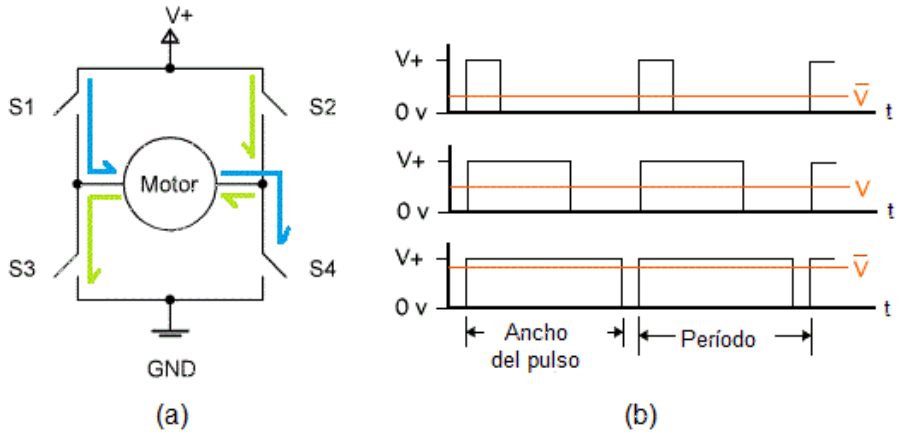


Figura 2.10: Control de sentido y velocidad de motores DC: (a) Inversión de giro al conmutar S1 y S4 o S2 y S3. (b) Variación de la velocidad mediante modulación de ancho de pulso.

2.3.4. Actuación sobre motores DC.

Para efectuar la inversión del giro o generar pulsos de tensión y corriente se requiere de interruptores electrónicos o switch. El switch electrónico por ex-

celencia es el relé, empleado en los primeros computadores, permite un aislamiento completo entre la señal efecto y la de control, sin embargo su lentitud y tamaño los condenan solo a aplicaciones de corriente alterna. Los transistores bipolares, conmutados entre etapas de corte y saturación, son switch electrónicos rápidos y eficientes, aunque no poseen total aislamiento cuando varía la estabilidad de la salida (por ejemplo el inductor de un motor). Los transistores CMOS, poseen mejor aislamiento, por lo que son los más comúnmente usados para los puentes H que controlan motores DC de gran potencia. Sin embargo, para que un arreglo con este tipo de transistores sea confiable, se requiere de etapas de estabilización y realimentación con componentes pasivos que hacen mucho más grande un el circuito estabilizado y confiable, como se muestra en la Figura 2.11, cabe destacar que el uso de transistores de tipo N y P tipo presentan retardos en el encendido que incide en un recalentamiento adicional del dispositivo.

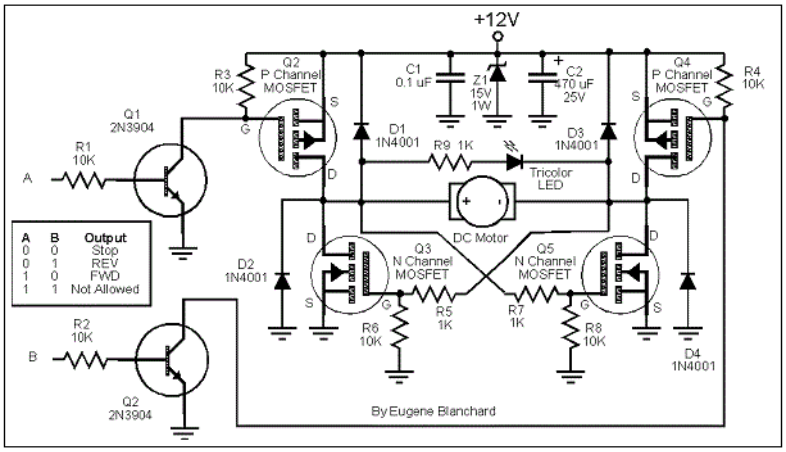


Figura 2.11: Puente H a transistores estabilizado.
Fuente: Eugene Blanchard (2007)[8]

Otra alternativa es el empleo de Triac's como interruptor controlado en DC. Para accionarlos se requeriría de optotriacs o de otros componentes para su activación, sin embargo es un dispositivo que se apaga cuando la corriente de carga se extingue por completo, y el bobinado inductor de los motores hace permanecer una corriente durante un período de tiempo, lo que hace necesaria la inversión del sentido de la tensión (y la corriente) para apagar la alimentación del motor y frenarlo.

Los motores DC están presentes en muchas aplicaciones de micromecánica, como impresoras y otros artefactos industriales, para lo cual existen circuitos integrados especializados para tal fin, entre ellos, los únicos que se consiguen en

el mercado venezolano son los integrados puente L298N, LMD18200T, y L6203, mostrados en la Figura 2.12, donde resumen sus principales características.

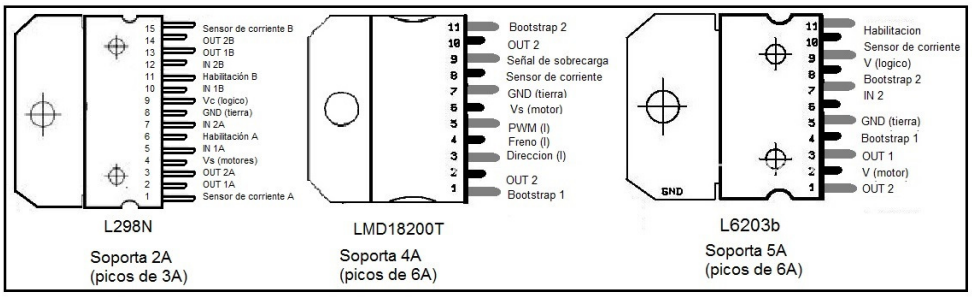


Figura 2.12: Circuitos integrados puente disponibles.

2.4. Elementos de hardware.

2.4.1. Optoacopladores.

Un opto acoplador, también llamado opto aislador o aislador acoplado ópticamente, es un dispositivo para la transmisión de información que funciona como un interruptor electrónico, funciona combinando un fototransmisor y un fotoreceptor en un solo encapsulado; suele ser un Diodo Emisor de Luz (LED) y un fototransistor, de esta manera la activación del transistor depende de la recepción de luz (únicamente proveniente del LED) y no de corrientes eléctricas susceptibles a fluctuaciones de voltaje. El funcionamiento es el mismo para este tipo de dispositivos, cambiando muy poco las características de un modelo a otro. Pueden señalarse los circuitos integrados PC817 y 4N35 mostrados en las Figuras 2.13 (a) y (b) donde se expone su constitución interna.

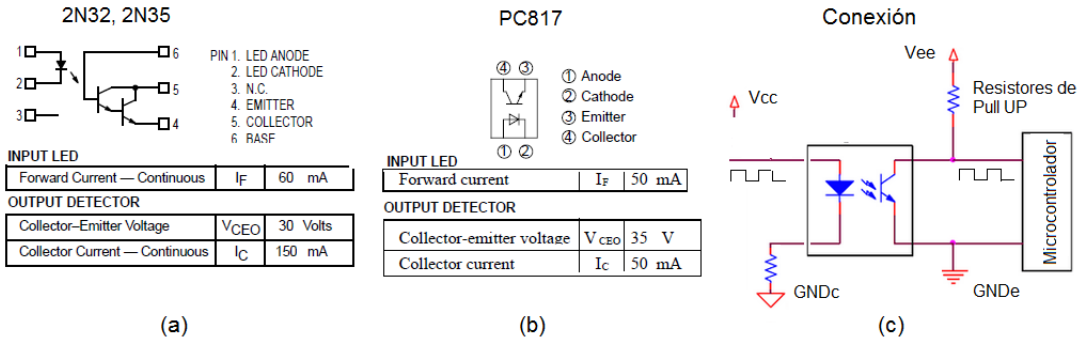


Figura 2.13: Opto-acopladores: (a) y (b) Integrados comunes (c) Conexión utilizada .

La conexión de estos dispositivos es variada, sin embargo se emplea comúnmente una salida en colector abierto, que consiste en la colocación de una resistencia denominada de "Pull UP" entre el colector del fototransistor y el nivel alto de alimentación, ver Figura 2.13 (c), lo que produce un nivel alto para una salida en el colector, que se vuelve nivel bajo (casi cero voltios) cuando el transistor es saturado por la luz recibida por el LED al encenderse. Como puede observarse en la mencionada figura estos dispositivos permiten separar los voltajes y corrientes de alimentación, recomendándose una misma referencia de "tierra" (GNDc =GNDe) único conductor al mismo potencial por el que no debe circular corriente pero si mantener la referencia de voltaje.

2.4.2. Multiplexores y demultiplexores en lógica TTL.

La denominación TTL (Transistor-Transistor Logic) proviene de la constitución interna del dispositivo a base de transistores, la tensión de alimentación característica de este tipo de dispositivos es Vcc= 5V ±5 %, donde se considera lógica binaria cero "0" a la tensión comprendida entre 0,0V y 0,8V; entrada binaria uno "1" entre 2,5V y Vcc. De todas las familias de dispositivos TTL, la más extendida es la familia LS (low power schottky) que es de bajo consumo empleando diodos schottky. DIP es un tipo de encapsulado para este tipo de dispositivos que emplea una separación estándar entre dos pines o terminales de 2,54 mm [9].

Son circuitos combinacionales que permiten seleccionar una entrada o una salida especifica según la lógica combinacional aplicada a sus pines de control. Un Multiplexor (mux) posee varias entradas y una única salida que será la entrada específica seleccionada, un demultiplexor (demux) por su parte no posee entradas, sino que coloca una salida activo bajo (lógica cero binario) según la selección de sus pines de control. Ambos circuitos integrados poseen entradas de habilitación que permiten desactivar las salidas del dispositivo.

Su implementación como circuitos integrados para los niveles lógicos TTL emplea la identificación 74, siendo el 74LS151 un multiplexor (o mux) y el 74LS138 un demultiplexor (demux), mostrados en la Figura 2.14, los cuales tienen tres entradas de control, tolerando una temperatura de 70°C, corriente de entrada máxima cercana a 1mA y poseen un retardo acotado en 50ns [10].

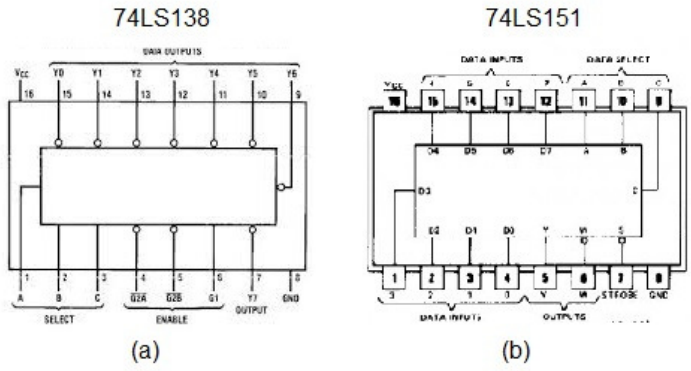


Figura 2.14: Multiplexación TTL: (a) Demultiplexor (b) Multiplexor. Fuente: Philips ECG Inc. (1987)[10]

2.4.3. Hardware programable

Un microcontrolador (MCU) es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales que cumplen tareas específicas. Existen muchos tipos de micro-controladores, sin embargo para muchas de las aplicaciones experimentales se suele usar una arquitectura basada en un único procesador con igual tiempo de ejecución para cada instrucción base en assembler.

Microchip Technology Inc. es una empresa fabricante de micro-controladores y otros dispositivos electrónicos. Sus micro-controladores pueden activar o desactivar los pines de uno de sus puertos bajo los niveles lógicos TTL; tolerando una corriente máxima de salida de 200mA en la mayoría de los casos. Poseen además útiles bloques funcionales como ADC, PWM, USB, UART entre otros. Este fabricante se distingue por incorporar las siglas PIC como encabezado del nombre de sus dispositivos, siendo las familias de dispositivos PIC16 y PIC18 las más frecuentemente usadas en implementación experimental de baja velocidad.

2.4.4. Programación de micro-controladores Microchip®

El procesador del microcontrolador ejecuta línea por línea las instrucciones cargadas en la memoria de programa. Para que el procesador sea capaz de interpretarlas, estas instrucciones deben ser escritas en un lenguaje llamado assembler, que es interpretativo y depende del fabricante, Microchip Technology ofrece gratuitamente MPLAB. Sin embargo existen software de computado-

ra que permiten traducir instrucciones lógicas más cercanas al pensamiento humano (alto nivel) a instrucciones equivalentes en lenguaje assembler, estos software se llaman compiladores y existen muchos en el mercado.

MikroC es un software compilador en lenguaje C desarrollado por la empresa Mikroelectronica; es importante destacar que así como el castellano presenta dialectos, los "lenguajes" de programación también, y varían según el software específico. De esta manera MikroC presenta una variación del "lenguaje C" específica, que puede ser estudiada mediante la ayuda proporcionada por el fabricante en el mismo software, escrita en forma autodidacta. De esta manera, la lógica programada en lenguaje C (en un archivo con extensión ".c") puede ser traducida a lenguaje assembler (archivo con extensión ".hex") que podrá ser cargado al microcontrolador.

Para transferir las instrucciones del archivo con extensión ".hex" al microcontrolador se requiere de un software y un circuito específicos (programador) que conectado al microcontrolador se le indique que será reprogramado. Microchip Technologies ofrece a la venta un programador conectado USB para estos dispositivos, aunque también expone gratuitamente a la venta el software Pickit2 y los planos del circuito específico compatible con él. Este artefacto permite programar los micro-controladores de las familias PIC12 PIC16 y PIC18 de este fabricante. El elemento más general de la gama PIC18 es el hardware que comparten PIC18F2550 (28pines) y el PIC18F4550 (40pines), por incorporar todas las ventajas mejoradas de la gama [7]. En la Figura 2.15 se muestra el esquema general de programación de un microcontrolador.

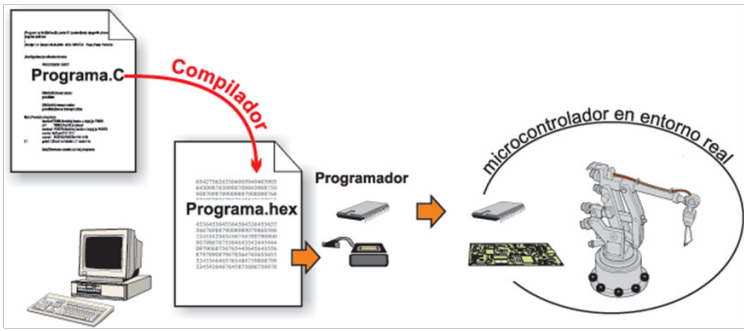


Figura 2.15: Esquema general de programación de un microcontrolador.
Fuente: González-Frassato (2014)[11]

Además de la programación general, el archivo de extensión .hex incluye las configuraciones base del dispositivo, las cuales determinan entre otras cosas la velocidad en que serán leídas las instrucciones, que dependen de un cristal

oscilador conectado externamente y puede tener distintos valores. A estas características de configuración base se le llama "palabra de configuración". Los software compiladores (convertidores de lógica a .hex) suelen proveer una ventana para esta configuración. En MikroC For PIC es llamada "Edit Project".

Para el soporte de datos del bus USB algunos microcontroladores poseen un hardware especializado en transferir los datos de alta velocidad (96MHz) a la memoria y sincronizar estas acciones con las realizadas por el procesador de actividad central. Para ello el microcontrolador debe configurarse con un oscilador interno de 48 MHz en conjunto con un post-escalador que la lleve a los 96MHz requeridos por el bus. El hardware se sincroniza con un reloj de 4 MHz (divisor entero de 96MHz), por lo cual, en caso de emplearse un cristal externo diferente a 4MHz, por ejemplo 20MHz, deberá usarse un preescalador que divida esta frecuencia para llevarla a 4Mhz.

2.4.5. Empleo del ADC y PWM con mikroC PRO for PIC.

La conversión analógica-digital (ADC, Analogic to Digital Converter) o digitalización consiste en la transcripción de señales analógicas en señales digitales, con el propósito de facilitar su procesamiento (codificación, compresión, etc.). Para ello se requieren dos etapas; la cuantización y la codificación, la primera de ellas consiste en la comparación de una señal analógica de entrada con una señal escalonada genera por el dispositivo a una velocidad mucho mayor tal que la señal de entrada no varíe en ese tiempo (muchas veces se promedian automáticamente conversiones para evitar errores). De esta manera el convertidor puede generar una señal escalonada de diferentes pisos y frecuencias de conversión, sujetas a un límite mínimo teórico de frecuencia denominado "Frecuencia de Nyquist" [12]. La codificación consiste en la transformación del valor digital que indica el escalón de la comparación en un número que el caso de los micro-controladores suele ser hexadecimal para una cantidad específica de bits.

Los micro-controladores Microchip pueden poseer varios ADC. Para su entendimiento considérese que posee discretizadores en diferentes pines, denominados "canales" (channels), pero solo un codificador para la configuración del ADC que posee un indicador de fin de la conversión. Para su manipulación la familia o gama PIC18 emplea los siguientes registros del microcontrolador (tomando como referencia el hardware del PIC18f4550 y PIC18f2550 [13]): ADCON0 para el control: indicar la activación, leer el indicador de término y

especificar el canal de lectura; ADCON1 para configurar el hardware, esto es, activación o no de los canales y el empleo o no de comparadores analógicos (empleados como indicadores de control paralelo a la lógica del controlador); ADCON2 para especificar los tiempos de adquisición de datos y la frecuencia de captura, pero también la alineación de los bits de conversión (izquierda o derecha de los 10 bits en los dos bytes de memoria, permitiendo reducir precisión a conveniencia).

El software MikroC for PIC posee una librería para el manejo de este convertidor [14]. La función ADC_Init() carga valores típicos de configuración de los registros antes descritos, y se sugiere emplear la configuración manual de ellos para una aplicación específica. ADC_Read(Channel) realiza con detenimiento la ejecución del sistema, dedicándose solo a la captura de la conversión analógico-digital del canal del número especificado hasta su culminación. La conversión suele ser bastante rápida.

La modulación de ancho de pulsos (PWM, Pulse Widh Modulate), puede ser empleada mediante la simple activación o desactivación de una salida del microcontrolador entre los niveles lógicos TTL desde cualquier pin del circuito integrado, requiriendo necesariamente que el procesador se dedique únicamente a esa tarea. Sin embargo Microchip desarrolló un hardware que genera PWM en paralelo con el proceso, empleando la comparación con un un temporizador y un registro precargado que determinan la duración del pulso en estado alto y la frecuencia. Esto se configura mediante los registros CCPXCON (donde el X indica el número, ya que puede poseer más de uno en pines específicos). El compilador MikoC for PIC ofrece una librería de funciones que permiten la configuración y activación del PWM: PWM1_Init(frecuencia, preescalador de tiempo) para configurar, PWM1_Start() y PWM1_Stop() para su activación [14].

2.4.6. Bootloader

El Bootloader es una gran ventaja para la programación, ensayo y pruebas de microcontroladores, ya que permite re-programar el microcontrolador sin necesidad de moverlo del protoboard o circuito impreso. Además, un bootloader USB permite verificar la confiabilidad del bus USB.

> Un gestor de arranque (en inglés “bootloader”), es un programa sencillo diseñado exclusivamente para preparar todo lo que nece-

> sita un computador para funcionar de una manera específica. Un microcontrolador tiene la misma arquitectura de un computador, por tanto se le llama bootloader al programa cargado en él, que sirve como plataforma para unas funciones base que permitan su programación sin necesidad de un hardware adicional como el Pikit2.

En este sentido la empresa MikroElectronica ofrece gratuitamente un bootloader, que consiste en un archivo con extensión ".hex" que debe ser programado en el microcontrolador por los medios tradicionales ocupando permanentemente ese espacio de la memoria de programa, esto permite que mediante un software llamado MikroBootloader ejecutado desde el computador y conectado vía USB con el chip, pueda ser escrito el resto de la memoria de programa sin necesidad de hardware especial.

Su forma de uso es el siguiente: se ejecuta la aplicación mikroBootloader, desplegando una ventana como la mostrada en la figura 2.16, se conecta el microcontrolador programado con el bootloader al puerto USB únicamente (desenergizado), se espera que el botón marcado con el número 1 se coloque en rojo y se pulsa (durará 5 segundos), luego se pulsa el botón 2, esto indica acceso al modo de programación, luego pulse el botón "3" y cargue el archivo con extensión ".hex" que desee programar en el microcontrolador, finalmente pulse el botón "4" y se programará.

Al ser energizado el dispositivo desde cualquier medio, el esperará cinco (5) segundos en espera de una señal USB del mikrobootloader, en caso de no recibirla, se ejecuta el resto del código programado en él.

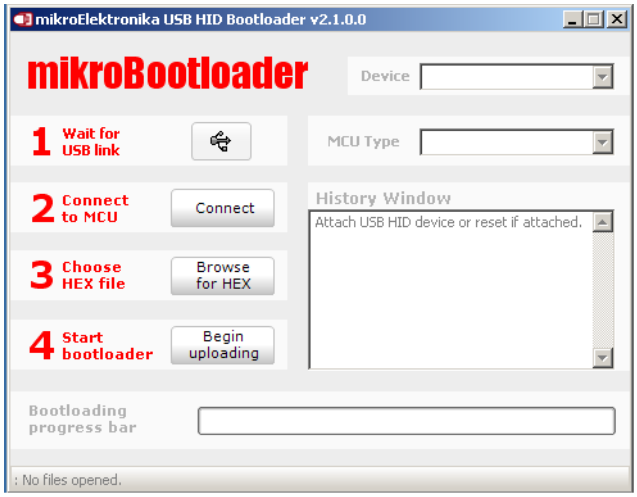


Figura 2.16: Ventana de Mikrobootloader

2.4.7. Simulación y diseño electrónico.

Existen muchos software que permiten el diseño, simulación y esquemas de implementación de circuitos electrónicos. Entre ellos existe “Proteus”, desarrollado por Labcenter Electronics que consta de los dos programas principales: Ares e Isis.

Isis (Intelligent Schematic Input System) permite diseñar el plano eléctrico del circuito que se desea realizar con componentes muy variados, desde simples resistencias, hasta alguno que otro microprocesador o microcontrolador, incluyendo fuentes de alimentación, generadores de señales y muchos otros componentes con prestaciones diferentes. Los diseños realizados en Isis pueden ser simulados en tiempo real, mediante el módulo VSM, asociado directamente con ISIS.

ARES (Advanced Routing and Editing Software); es la herramienta de enrutado, ubicación y edición de componentes, se utiliza para la fabricación de placas de circuito impreso, permitiendo editar generalmente, las capas superficial (Top Copper), y de soldadura (Bottom Copper).

2.5. Universal Serial Bus (USB)

Es un bus estándar industrial que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre computadoras, periféricos y dispositivos electrónicos (ver Figura 2.17). Al estandarizar la comunicación entre dispositivos, diferentes equipos pueden comunicarse sin necesidad de instaladores o “Drivers” (es el análogo humano a la unificación del idioma, tono de voz y modales de dialogo). Por esto se ha hecho muy popular en computadores personales desplazando a todos los demás puertos (ver Figura 2.17). La implementación de cualquier dispositivo moderno de interfaz cableada controlada por computador, requiere soportar la comunicación mediante el estándar universal USB.

2.5.1. Principales características del USB 2.0.

El estándar emplea cuarto hilos: Dos de alimentación, y dos de comunicación(+5v, tierra, D+ y D-), para lograr una comunicación Half Duplex, (ver Figura 2.18). Emplea los niveles lógicos de ±400 mV en modo diferencial para

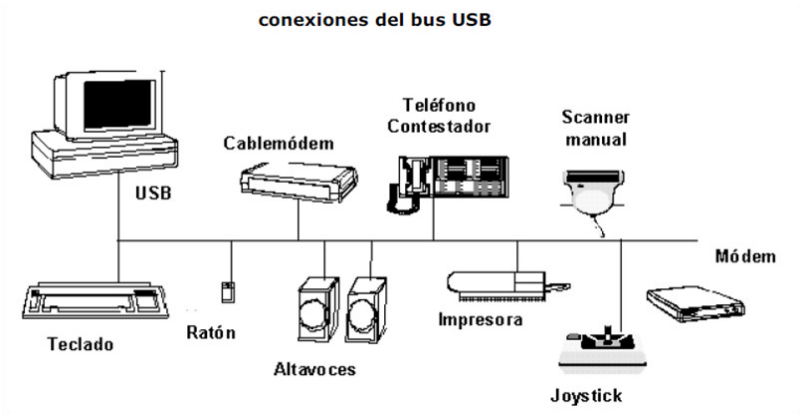


Figura 2.17: Variedad de dispositivos pueden ser conectados al bus USB.
Fuente: Jan Axelson-USB Complete (2001)[15]

reducir el efecto electromagnético; también posee una terminación de 45 Ω a tierra para acoplar la impedancia del cable, con lo que se logra comunicación confiable en alta velocidad hasta de 5 metros. Con respecto a la potencia, este puerto sólo admite la conexión de dispositivos de bajo consumo, es decir, que tengan un consumo máximo de 100 mA por cada puerto o extensión de puertos (con máximo de 5).

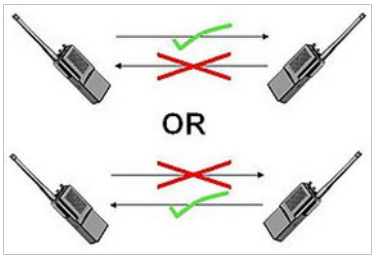


Figura 2.18: Comunicación Half Duplex, solo admite una transmisión por vez para un único sentido seleccionado.
Fuente: Floyd Thomas (2004)[9]

2.5.2. Interfaz física

La interfaz física esta formada por cuatro hilos, dos para la alimentación 5v (Rojo) GND (Negro) , y dos para datos D+ (Verde) y D- (Blanco) (ver Figura 2.19). Los signos + y - se utilizan para hacer referencia a que la señal de trabajo es diferencial cuyo valor depende la velocidad del bus.

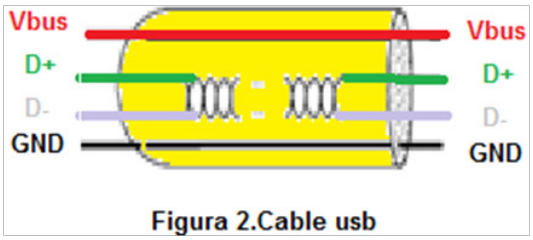


Figura 2.19: Estructura del cable USB.
Fuente: Daniel Ordoñez (2012)[16]

2.5.2.1. Conectores.

Los conectores internos a los equipos se establecieron conectores hembra, siendo unidos por el conector doble macho correspondiente. A diferencia de otros cables de datos (Ethernet, HDMI, etc), un cable USB fue diseñado para que cada extremo utilice un tipo de conector diferente (tipo A o tipo B), con el fin de incorporar protecciones eléctricas para el flujo de energía del computador a los periféricos, siendo el conector A hembra el puerto del computador. Algunos equipos de bajo consumo eléctrico y velocidad tienen incorporado un conector A macho únicamente, además existen otros tamaños "mini" y "micro" empleados comúnmente en cámaras y teléfonos (ver Figuras 2.20 y 2.21).

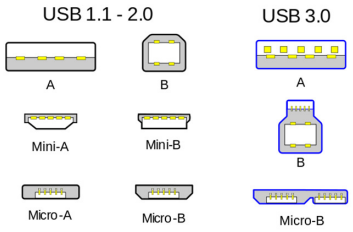
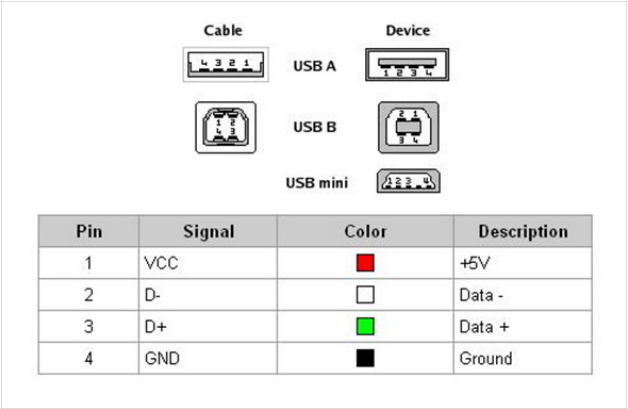


Figura 2.20: Diferentes conectores para el bus USB. Ambos tienen su versión hembra y macho.
Fuente: Jan Axelson-USB Complete (2001)[15]

2.5.3. Velocidades del Bus

Los dispositivos USB se clasifican en cuatro tipos según su velocidad de transferencia de datos (ver Cuadro 2.3):

Los computadores suelen incluir dos de las tres versiones de USB, lo que hace importante para la implementación de un hardware propio, el reconocer en



Pin	Signal	Color	Description
1	VCC		+5V
2	D-		Data -
3	D+		Data +
4	GND		Ground

Figura 2.21: Identificación de hilos del bus USB de los Tipos A y B
Fuente: Jan Axelson-USB Complete (2001)[15]

Cuadro 2.3: Tipos de dispositivos USB según velocidad de transferencia

Versión	Velocidad	Uso Común
1.0	Baja velocidad: hasta 1,5 Mbit/s (188 kB/s)	Dispositivos de interfaz humana: teclados, mouse, cámaras web, etc.
2.0	Alta Velocidad: máxima práctica de 280 Mbit/s (35 MB/s).	Casi todos los dispositivos fabricados en la actualidad trabajan a esta velocidad.
3.0	Superalta velocidad: hasta 4,8 Gbit/s (600 MB/s)	Permite soporte de datos HID. en 2009 comenzó a incluirse junto con el USB 2.0.6 7.

forma automática o manual los puertos correctos para asegurar la compatibilidad ante las diferentes velocidades de transmisión de datos en el puerto del computador.

2.5.4. Tipos básicos de transferencias

Una transferencia se puede definir como el conjunto global de los datos que forman una comunicación USB, una transferencia está formada a su vez por una o varias transacciones que están constituidas por diferentes paquetes de datos que contienen las tramas de una comunicación USB. No existe un formato único de transferencia, la especificación USB permite cuatro tipos de transferencias:

- Control: se utilizan para configurar y enviar comandos, por ejemplo en la enumeración del dispositivo.

- Bulk (masivas): Se utilizan cuando se precisa una transferencia de datos grande, es el tipo más rápido de transferencia, sin embargo no hay garantía de que los datos se transmitan en un tiempo determinado (no garantizada la latencia). Si hay verificación de que los datos se han transmitido con éxito ya que dispone de sistema de corrección de errores (CRC). Esta transferencia solo la pueden utilizar dispositivos que soporten velocidades Full y High Speed, como por ejemplo discos duros, escaners, impresoras, etc.
- Isócronas: Es usada en dispositivos que transmiten señales de audio y de vídeo en tiempo real. Se garantiza una tasa de velocidad de transmisión determinada (latencia asegurada). Si no fuese así, por ejemplo el audio de una transmisión de voz se oiría entrecortado. No contempla la corrección de errores, si en un archivo de sonido se pierde un bit, no es importante su recuperación. Para usar este tipo de transferencia es necesario que los dispositivos soporten velocidades Full Speed.
- Interrupción: Latencia asegurada y verificación de que los datos se han transmitido con éxito, Se utiliza en dispositivos como: teclados, mouse, sensores, pantallas táctiles, y dispositivos que no requieran mucho ancho de banda.

2.5.5. Definiciones para la comunicación estándar USB

2.5.5.1. Enumeración

El Host es el encargado de detectar cualquier dispositivo que se conecta al bus. Cuando un dispositivo es detectado el Host necesita obtener información sobre él, a este proceso se le llama enumeración. Esta información que necesita el Host se encuentra definida en el dispositivo en los denominados descriptores. Los descriptores son datos que se guardan en la memoria no volátil del PIC y contienen la siguiente información: El ID del vendedor (VID) y del producto (PID), consumo de corriente del dispositivo, tipo de transferencia que se va ha utilizar, endpoint utilizados, versión USB soportada, clase utilizada, etc. El VID (Vendor ID) y el PIC (Product ID) son dos números de 16 bits representados en Hexadecimal, propios de cada dispositivo.

2.5.5.2. Endpoint

Los endpoint son buffer de memoria RAM que son utilizados para el envío y recepción de datos o comandos de control durante una comunicación USB. Cada endpoint puede ser de entrada o salida de datos o bidireccional. El endpoint 0 está reservado para comandos de control, el proceso de enumeración se realiza a través del endpoint número 0. Este concepto solo se aplica al dispositivo, en el host existen también buffer para el envío y recepción de datos pero no se les denomina con este nombre.

2.5.5.3. Pipe

Es una conexión lógica entre un endpoint y el software del controlador del Host que se produce tras el proceso de enumeración. Los pipes se usan mucho en Sistemas Operativos como UNIX/LINUX para enlazar la salida de un proceso con la entrada de otro, en este caso el concepto es el mismo.

2.5.5.4. Clase

Una clase es un modelo o plantilla que describe el estado y el comportamiento de los objetos que la comparten. La clase provee de propiedades y métodos (funciones) reutilizables por los objetos o miembros que comparten la clase. La especificación USB provee de propiedades y funciones que pueden ser utilizadas por los dispositivos que tengan características similares. Por ejemplo, un teclado y un ratón por sus características pertenecerán a la misma clase la llamada Human Interface Device (HID). Una ventaja de utilizar esta clase por ejemplo es que no se necesita instalar ningún driver para el dispositivo ya que el sistema operativo utilizará uno genérico para todos.

Las clases más utilizadas con Microcontroladores son:

- HID (Human Interface Device): ejemplos de dispositivos que utilizan esta clase son: teclados, ratones, pantallas táctiles, joystick, etc. Velocidad low-speed (64 KB/s de velocidad máxima), tipos de transferencias soportadas: de control y de Interrupción. Una característica interesante al utilizar esta clase es que no se necesita instalar un driver especifico en el Sistema Operativo, se utiliza uno estándar que ya está incluido en el

sistema. En Windows la aplicación de escritorio que accede al dispositivo lo hace con ayuda de las librerías HID API, en las cuales se profundizará posteriormente.

- MSD (Mass Storage Device Class): Como su propio nombre indica para dispositivos de almacenamiento masivo como discos duros, memorias flash, cámaras digitales, dispositivos ópticos externos como lectores y grabadoras de CD y DVD, etc. Esta clase se puede utilizar solo en dispositivos que soporten velocidades Full y High Speed. El tipo de transferencias utilizadas es Bulk o una combinación formada por transferencias del tipo Control, Bulk y Interrupt.
- CDC (Communications Device Class): Un ejemplo de dispositivo que utiliza esta clase son los Modems, en este primer ejemplo utilizaremos esta clase para comunicar nuestro PIC18F4550 con la aplicación de escritorio que realizaremos con un IDE multiplataforma, que nos permitirá ejecutar la aplicación en diferentes sistemas operativos, concretamente en Windows, Linux y MAC. La velocidad máxima al utilizar esta clase será de 80 kBytes/s y el tipo de transferencias soportadas son del tipo interrupción y Bulk [17].

2.5.6. Liberia para recepción de datos en el computador HID API

HID API es una biblioteca multiplataforma que le permite a una aplicación interactuar con USB y dispositivos Bluetooth de clase HID en Windows, Linux y Mac OS X. Se puede utilizar para comunicarse con dispositivos HID estándar como teclados, ratones y joysticks. Es más útil cuando se utiliza con el modo personalizado (definido por el proveedor) de los dispositivos HID. Muchos dispositivos hacen esto con el fin de no requerir un controlador personalizado para ser escrito para cada plataforma. HID API es fácil de integrar con la aplicación cliente, sólo requieren un único archivo usado como biblioteca desde la aplicación. En Windows, HID API opcionalmente se puede construir en un archivo DLL. Los programas que utilizan HID API son sin controlador, lo que significa que no requieren el uso de un controlador personalizado para cada dispositivo en cada plataforma y en el caso de Windows, utiliza el Windows HID API que está implementado en todas las versiones del sistema operativo desde XP, por lo que se garantiza una compatibilidad prolongada en la plataforma. HID

API proporciona una interfaz limpia y consistente para cada plataforma, por lo que es más fácil desarrollar aplicaciones que se comunican con dispositivos HID USB sin necesidad de conocer los detalles de las bibliotecas HID y las interfaces de cada plataforma.

2.5.6.1. Funciones mas relevantes de la librería HID API

HID_Init

`hid_init(void)`: Libera todos los enlaces con dispositivos con el fin de iniciar o reiniciar la interfaz HID.

HID_Enumerate

`hid_enumerate(unsigned short vendor_id, unsigned short product_id)`: Esta regresa un objeto que representa un puntero de la lista de los seriales de dispositivos que coincidan con el Vendor ID y el Product ID especificados, en caso de que ambos valores sean "0", se devuelve la lista de todos los dispositivos. Dicho puntero, se utilizará para luego abrir una conexión con el dispositivo de interés.

HID_Open

`hid_open(unsigned short vendor_id, unsigned short product_id, wchar_t *serial_number)`: La función abre un canal de comunicación entre la aplicación y el dispositivo, devuelve un puntero identificador de dispositivo que posteriormente será utilizado para realizar la comunicación con las demás funciones.

HID_Write

`hid_write(hid_device *device, const unsigned char *data, size_t length)`: Esta función recibe el puntero del dispositivo previamente obtenido con HID_Open, el arreglo de datos y el tamaño del mismo y se encarga de enviar dichos datos al dispositivo. Dicha función devuelve el número de bytes escritos al dispositivo o "-1" en caso de que haya ocurrido un error.

HID_Read_Timeout

`hid_read_timeout(hid_device *dev, unsigned char *data, size_t length, int milliseconds)`: Al igual que `HID_Write`, esta función recibe el puntero del dispositivo, un puntero al arreglo de datos de salida, el número de datos a leer

y el tiempo de Timeout. Esta lee los datos recibidos de un dispositivo HID especifico, devuelve el número de bytes recibidos y en caso de pasar un tiempo especificado sin recibir datos, la función devuelve "-1" .

HID_Close

`hid_close(hid_device *device)`: Esta función simplemente cierra la conexión entre el dispositivo especificado y la aplicación.

HID_Exit

`hid_exit(void)`: Esta función cierra todos los enlaces hechos por HID y cierra la biblioteca [18].

2.5.7. La implementación con micro-controlador.

Las velocidades del bus, muy superiores a las velocidades típicas de los procesadores de Microhip, requieren de un hardware especializado presente en algunos dispositivos de la familia PIC18. Este hardware trabaja en paralelo con el procesador a una mayor velocidad empleando el ciclo de reloj impuesto dentro del propio bus, de esta manera interrumpirá el proceso principal solo cuando se termine la transferencia del paquete completo de datos (varios bytes), mediante una bandera de interrupción. Para lograr esto, existen registros de memoria especiales dentro del microcontrolador a los que se puede acceder tanto con el procesador como por el hardware especializado USB.

Debido a la complejidad del protocolo de comunicación USB La mayoría de los compiladores para PIC18 poseen librerías de funciones para el manejo de datos y configuración del bus. "MikroC PRO for PIC" puede ser usado tanto en el modo HID (de conexión continua con el dispositivo hasta fin de la transferencia de datos), como el modo "Polling" (donde el envío-recepción de paquetes se realiza por suceso de eventos). El modo HID se habilita con la función `HID_Enable(Br,Bw)`, donde debe indicarse los apuntadores a las direcciones de memoria específicos donde se recibirán los paquetes: `Br` para lectura y `Bw` para escritura. La línea de instrucción "`n = HID_Read()`", retornará al valor "`n`" el número de caracteres recibidos, cargando estos en el buffer de lectura del microcontrolador (`Br`). Para la escritura debe cargarse el paquete en la dirección de memoria `Bw` y dar la instrucción `HID_Write(*Br,cantidad_de_bits)` donde se indica el apuntador de la dirección donde está el byte a enviar y el tamaño que deberá leer.

La comunicación por USB requiere que el computador reconozca al dispositivo para saber cómo referirse a él, para ello el circuito deberá informar su identificación y el computador deberá ser capaz de leerlo. En el primer caso se requiere de un archivo que contenga los "descriptores", el cual debe ser compilado junto con el código fuente al momento de generar el archivo .hex. Debido a que deben ser ubicados en lugares específicos de la memoria que el compilador sabe ubicar, este archivo contendrá la siguiente información: identificador de producto y vendedor, tamaño de los paquetes de envió y recepción, corriente que el computador transferirá en mili-amperios y tiempo de espera tras cada subpaquete. Estas indicaciones básicas serán enviadas al computador como pautas para la comunicación con este dispositivo particular. El computador por su parte requiere poseer el software compatible con las especificaciones técnicas del USB que empleó el compilador para ese paquete de datos recibido al inicio de la comunicación y sin el cual los equipos no podrían saber como enviar o recibir sus datos. Por lo general vienen pre-cargados por defecto o se pueden instalar software que permitan esto. El propio MikroC tiene una herramienta llamada HID Terminal (en la pestaña Tools) que provee una interfaz para transmisión de paquetes de datos por USB.

2.5.8. Implementación USB con aplicaciones sobre Windows.

Para acceder a los recursos desde una aplicación propia en algún software genérico que se desee utilizar, se requiere por lo general de un componente de software que para ambiente Windows es llamado "Microsoft .NET Framework", el cual es incluido en los sistemas operativos Microsoft Windows. Debe tomarse en cuenta que la aplicación que se desarrolle debe ser compatible con la version del Framework del equipo. Este componente provee de soluciones pre-codificadas los requerimientos más comunes de los programas y gestiona la ejecución de ellos, ya que Microsoft desea que todas las aplicaciones creadas para la plataforma Windows sean basadas en el .NET Framework. Este requerimiento puede ser descargado gratuitamente desde la página web oficial de Microsoft. Su objetivo es estandarizar el desarrollo de software sencillo, reduciendo las vulnerabilidades y aumentando la seguridad de los programas desarrollados. El Framework incluye soluciones en áreas como: acceso a datos, la interfaz de usuario, conectividad a bases de datos, criptografía, desarrollo de aplicaciones web, algoritmos numéricos y comunicación de redes.

2.6. Cinemática de Robots Industriales

2.6.1. Representación de la orientación de un sólido

En el caso de un punto bastan tres coordenadas para determinar su posición en el espacio pero un sólido se compone por una multitud de puntos con una geometría particular, por lo que se hace necesario recurrir a otra característica que defina su orientación espacial: la orientación. Como orientación puede entenderse la representación respecto a un sistema de referencia fijo (OXYZ) de un grupo de tres componentes o grados de libertad definidos respecto a un sistema de referencia móvil (O'UVW) unido sólidamente al objeto. Las tres componentes referidas se identifican mediante tres vectores unitarios pertenecientes al sistema móvil (Figura 2.22), ellos son:

- El approach($\overrightarrow{a}$): indica la forma en que la herramienta "ataca" o enfrenta el plano de trabajo. Se encuentra definido en la dirección del eje W.

- El slide ($\overrightarrow{s}$): indica el sentido de trabajo de la herramienta. Se encuentra definido en la dirección del eje V. En el caso de "dedos" como los mostrados en la Figura 2.22, indica el sentido de apertura y cierre.

- El vector normal($\overrightarrow{n}$): completa el sistema dextrógiro de vectores unitarios en el sistema de referencia móvil. Se encuentra definido en la dirección del eje U [1].

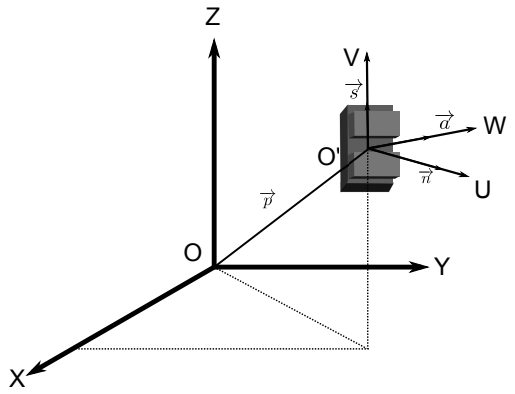


Figura 2.22: Vectores: normal slide y approach
Fuente: W. Sanz (2009)[1]

2.6.2. Matrices de rotación

Para modelar la orientación se partirá de una condición inicial donde los sistemas de referencia, móvil y fijo, tienen plena coincidencia entre sus ejes; luego se procede a girar el sistema móvil, un ángulo θ, alrededor del eje Z. O'. Una vista de planta de la condición mostrada en la Figura 2.23 permite analizar el problema en el plano $XUYV$:

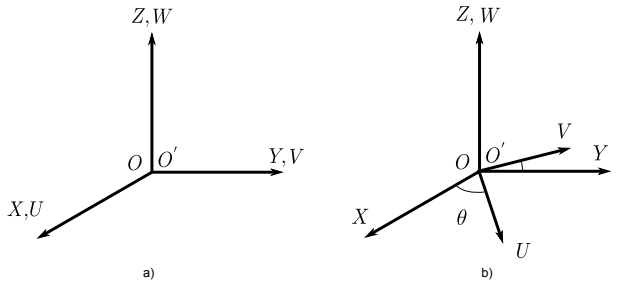


Figura 2.23: a) Condición inicial de coincidencia entre sistemas de referencia móvil y fijo. b) Giro del sistema de referencia móvil alrededor del eje Z
Fuente: W. Sanz (2009)[1]

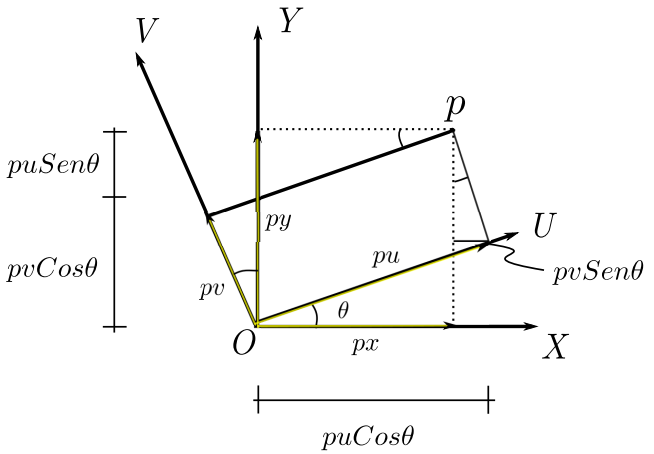


Figura 2.24: Análisis de una rotación pura en el plano $XUYV$
Fuente: W. Sanz (2009)[1]

A partir de la figura 2.24 son fácilmente observables las siguientes relaciones:

$$px = puCos\theta - pvSen\theta$$
$$py = puSen\theta - pvSen\theta$$

Las expresiones anteriores pueden reescribirse en forma matricial, como sigue:

$$\begin{bmatrix} px \\ py \end{bmatrix} = \begin{bmatrix} Cos\theta & -Sen\theta \\ Sen\theta & Cos\theta \end{bmatrix} = \begin{bmatrix} pu \\ pv \end{bmatrix}$$

La matriz de 2×2 que está en el centro de la ecuación previa se conoce como Matriz de Rotación (R). Ella describe las variaciones de la orientación del sistema de referencia móvil $0UV$ respecto al sistema de referencia fijo $0XY$. Una expresión más general de esta matriz surge al relacionar sus elementos con los productos escalares de los vectores unitarios correspondientes a los ejes de los sistemas involucrados. A continuación se muestran a estos vectores y a los ángulos que existen entre ellos.

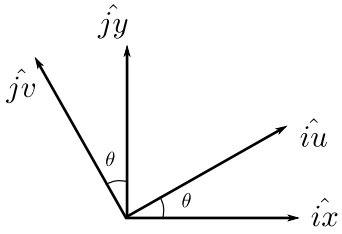


Figura 2.25: Vectores unitarios del plano $XUYV$
Fuente: W. Sanz (2009)[1]

Resultado de los productos escalares que se plantean a partir de la Figura 2.25, se obtiene la siguiente matriz de rotación:

$$R = \begin{bmatrix} \hat{ix} \bullet \hat{iu} & \hat{ix} \bullet \hat{jv} \\ \hat{jy} \bullet \hat{iu} & \hat{jy} \bullet \hat{jv} \end{bmatrix}$$

A partir de esta expresión más general puede deducirse lo que podría identificarse como una "regla de formación" que permite crear una Matriz de Rotación para el espacio euclídeo:

$$R = \begin{bmatrix} \hat{ix} \bullet \hat{iu} & \hat{ix} \bullet \hat{jv} & \hat{ix} \bullet \hat{kw} \\ \hat{jy} \bullet \hat{iu} & \hat{jy} \bullet \hat{jv} & \hat{jy} \bullet \hat{kw} \\ \hat{kz} \bullet \hat{iu} & \hat{kz} \bullet \hat{jv} & \hat{kz} \bullet \hat{kw} \end{bmatrix} \tag{2.1}$$

La ecuación (2.1) se conoce como Matriz de Cosenos Directores o Matriz de Rotación. Ella describe las variaciones en la orientación de un sistema de referencia móvil $0UVW$ respecto a un sistema de referencia fijo $0XYZ$. [1]

2.6.3. Transformaciones homogéneas

Se define como localización a la representación conjunta de la posición y la orientación de un sólido en el espacio. Esto puede lograrse mediante un sistema de coordenadas llamado Homogéneo, el cual representa a un espacio de n dimensiones en $(n+1)$ coordenadas que fue ideado por el matemático alemán August Ferdinand Mobius.

En el caso de vectores la representación en el espacio homogéneo puede asumirse como la adición de una coordenada extra, llamada factor de escala, la cual debe multiplicar a todas sus componentes. el vector factor de escala es arbitrario, por lo que es práctica común elegirlo igual a uno por simplicidad.

En el caso de matrices la representación en el espacio Homogéneo se logra aumentando en uno las dimensiones de la matriz. en Robótica este aumento es intencionado para lograr la combinación de las transformaciones que describen los procesos de rotación y traslación. El resultado es una nueva matriz llamada Matriz de Transformación Homogénea (T), que incluye una submatriz para describir la orientación (Matriz de Rotación), la posición (Vector de traslación), la perspectiva (vector nulo, por simplicidad) y el factor de escalado (igual a 1).

$$T = \begin{bmatrix} Rotación_{3\times3} & Traslación_{3\times1} \\ Perspectiva_{1\times3} & Escalado_{1\times1} \end{bmatrix} \tag{2.2}$$

Para un vector cualquiera, representativo de un punto en el espacio, una matriz de Transformación Homogénea permite obtener el resultado de su rotación y traslación conjuntas; lo cual es completamente pertinente para los estudios que se realizan en esta área de la Robótica.[1]

2.7. Estudio Cinemático del Robot XR-3

2.7.1. Cinemática Directa

A través de este estudio se busca conocer la posición final del elemento terminal de un Robot Manipulador en relación con un sistema de referencia fijo, sobre la base del conocimiento previo de sus articulaciones y los parámetros geométricos de sus elementos.

El punto final de un eslabón articulado puede realizar un movimiento respecto al eslabón anterior susceptible de descripción mediante una matriz de Transformación Homogénea (T), esta es la matriz que se debe hallar para obtener el resultado de las rotaciones o traslaciones efectuadas por cada articulación.

Para el robot Rhino XR-3 que posee 5 articulaciones y 6 elementos (incluyendo la base), la posición del elemento final referida a la base, se describe mediante un conjunto de post-multiplicaciones matriciales equivalentes a una sola matriz de Transformación (T):

$$T = A_1^0 * A_2^1 * A_3^2 * A_4^3 * A_5^4 \tag{2.3}$$

Se utiliza la letra A para referirse al movimiento relativo entre eslabones, posee un súper índice que indica al eslabón que sirve de base al movimiento relativo entre dos elementos, y un sub-índice para señalar al eslabón que realiza el movimiento. Para obtener la cadena cinemática que caracterice a este robot manipulador a través de la relación entre sus elementos y articulaciones, se debe aplicar un método de obtención sistemática de la situación del efector final respecto a un sistema de referencia fijo, conocido como Algoritmo de Denavit-Hartenberg (D-H). [1]

En la Figura 2.26 se muestra el modelo funcional y sistemas de coordenadas, lo que permite describir el movimiento relativo de sus eslabones a través de los ya mencionados parámetros pertenecientes al Algoritmo D-H.

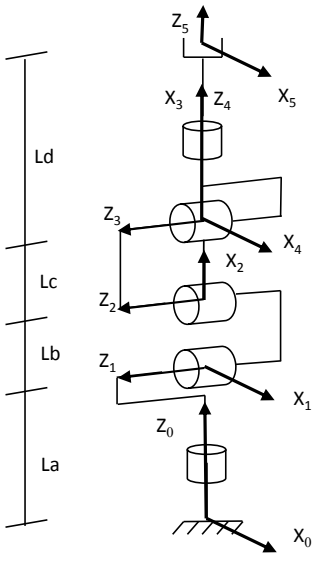


Figura 2.26: Sistema de Coordenadas Generalizadas
Fuente: W. Sanz (2009) [1]

2.7.1.1. Matrices de paso Homogéneas

Describen el movimiento relativo entre dos eslabones de una cadena cinemática. Permiten relacionar al sistema de referencia del elemento " i " con el del elemento " i-1 ". La localización de un punto del elemento " i " (asociado a un sistema tomado como móvil), puede definirse respecto al sistema de elemento previo " i-1 ", tomado como sistema fijo, mediante una Transformación homogénea de cuatro eventos: rotación alrededor del eje Z_{i-1} en un angulo θ_i; traslación a lo largo del Z_{i-1}, un distancia di; traslación a lo largo del eje $X_i$una distancia ai y finalmente una rotación alrededor del eje X_i un angulo α_i.

$$A_i^{i-1} = T(z, \theta_i)T(0, 0, di)T(ai, 0, 0)T(x, \alpha)$$

La ecuación general de la Matriz de Transformación del Robot es:

$$A_i^{i-1} = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & aC\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & aS\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

En el Cuadro 2.4 se identifica el número de articulación y el valor del parámetro correspondiente. El parámetro "a" es la distancia normal que existe entre los ejes articulares Z_{i-1} y Z_i con respecto al eje X_i , el parámetro "α" es el ángulo de torsión que existe entre ejes Z_{i-1} y Z_i con respecto al eje X_i, el parámetro "θ" representa la distancia angular que existe entre los eje normales X_{i-1} y X_i con respecto al eje Z_{i-1}, siendo este ángulo variable en las articulaciones rotoides y constante en las articulaciones prismáticas, y por último el parámetro "d" representa la distancia lineal que existe entre los ejes normales X_{i-1} y X_i con respecto al eje Z_{i-1}, el cual permanece constante en articulaciones rotoides y variable en las articulaciones prismáticas.

Las matrices de transformación particulares para el caso de estudio son:

$$A_1^0 = \begin{bmatrix} C1 & 0 & S1 & 0 \\ S1 & 0 & -C1 & 0 \\ 0 & 1 & 0 & La \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2^1 = \begin{bmatrix} C2 & -S2 & 0 & LbC2 \\ S2 & C2 & 0 & LbS2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Articulación	θ	d	a	α	q
1	q_1	La	0	90	θ_1
2	q_2	0	Lb	0	θ_2
3	q_3	0	Lc	0	θ_3
4	q_4	0	0	-90	θ_4
5	q_5	Ld	0	0	θ_5

Cuadro 2.4: Parámetros DH

$$A_3^2 = \begin{bmatrix} C3 & -S3 & 0 & LcC3 \\ S3 & C3 & 0 & LcS3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_4^3 = \begin{bmatrix} C4 & 0 & -S4 & 0 \\ S4 & 0 & C4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5^4 = \begin{bmatrix} C5 & -S5 & 0 & 0 \\ S5 & C5 & 0 & 0 \\ 0 & 0 & 1 & Ld \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

donde se ha realizado la sustitución $Cos(\theta_n) = Cn \quad Sen(\theta_n) = Sn$.

2.7.1.2. Elementos de la matriz de Transformación del Robot (T)

La Matriz de Rotación está compuesta de tres vectores que son usados para modelar la rotación del TCP con respecto a un sistema de coordenadas móvil, los cuales son: el vector Normal $\vec{n}$:(n_x, n_y, n_z), el vector *Slide* $\vec{s}$:(s_x, s_y, s_z) y el vector *Approach* $\vec{a}$:(a_x, a_y, a_z) y para la posición del TCP con respecto al sistema de referencia escogido para la base del robot, el vector de traslación $\vec{p}$:(p_x, p_y, p_z). Quedando así del producto de las matrices de Paso Homogéneas, la matriz T. Dado su tamaño se muestran a continuación sus elementos [1].

$$T = \begin{bmatrix} nx & sx & ax & px \\ ny & sy & ay & py \\ nz & sz & az & pz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Las ecuaciones que conforman la Matriz de Transformación del Robot son:

$$
\begin{aligned}
nx =&((C1C2C3 - C1S2S3)C4 - (C1C2S3 + C1S2C3)S4)C5 - S1S5\\
ny =&((S1C2C3 - S1S2S3)C4 - (S1C2S3 + S1S2C3)S4)C5 + C1S5\\
nz =&((S2C3 + C2S3)C4 + (-S2S3 + C2C3)S4)C5
\end{aligned}
$$

$$
\begin{aligned}
sx =& - ((C1C2C3 - C1S2S3)C4 - (C1C2S3 + C1S2C3)S4)S5 - S1C5\\
sy =& - ((S1C2C3 - S1S2S3)C4 - (S1C2S3 + S1S2C3)S4)S5 + C1C5\\
sz =& - ((S2C3 + C2S3)C4 + (-S2S3 + C2C3)S4)S5
\end{aligned}
$$

$$
\begin{aligned}
ax =& - (C1C2C3 - C1S2S3)S4 + (-C1C2S3 - C1S2C3)C4\\
ay =& - (S1C2C3 - S1S2S3)S4 + (-S1C2S3 - S1S2C3)C4\\
az =&(S2C3 + C2S3)S4 + (-S2S3 + C2C3)C4
\end{aligned}
$$

$$
\begin{aligned}
px =& [-(C1C2C3 - C1S2S3)S4 - (C1C2S3 + C1S2C3)C4]\, Ld\\
&+ C1C2LcC3 - C1S2LcS3 + C1LbC2
\end{aligned}
$$

$$
\begin{aligned}
py =& [-(S1C2C3 - S1S2S3)S4 - (S1C2S3 + S1S2C3)C4]\, Ld +\\
&+ S1C2LcC3 - S1S2LcS3 + S1LbC2
\end{aligned}
$$

$$
\begin{aligned}
pz =& [-(S2C3 + C2S3)S4 + (-S2S3 + C2C3)C4])Ld\\
&+ S1C2LcC3 - S1S2LcS3 + LbS2 + La
\end{aligned}
$$

2.7.2. Cinemática Inversa

La Cinemática inversa busca determinar el conjunto de valores de los parámetros θ y d de un robot, que satisfacen la necesidad de orientar y posicionar su herramienta (TCP) en un punto, de un modo específico. Pueden existir distintas trayectorias o caminos que puedan posicionar al TCP en el lugar requerido,

o quizás no exista ninguna forma en que pueda lograrse.

Para obtener las soluciones particulares que resuelvan la cinemática inversa, es posible apelar a estudios gráficos y trigonométricos o bien utilizar procedimientos analíticos. La primera opción, requiere gran capacidad de observación y ciertas aptitudes como dibujante. Usarla resulta conveniente cuando se trata de robots de pocas articulaciones y configuraciones sencillas.

Con la utilización de procedimientos analíticos, como el modelo matemático que se obtiene como resultado del Algoritmos de Denavit-Hartenberg para dar solución al problema cinemático directo, es posible en algunos casos utilizar el conjunto de ecuaciones resultante para despejar los parámetros θ y d. El objetivo es plantear un sistema con igual número de ecuaciones que de incógnitas (θ y d), lo cual puede alcanzarse mediante el desarrollo de pre multiplicaciones del tipo generalizado en la Ecuación 2.4.

$$\begin{aligned} T = A_1^0 * A_2^1 * ... * A_n^{n-1} \\ (A_1^0)^{-1} * T = A_2^1 * A_3^2 ... * A_n^{n-1} \\ (A_2^1)^{-1} * (A_1^0)^{-1} * T = A_3^2 ... * A_n^{n-1} \\ (A_{n-1}^{n-2})^{-1} * ... * (A_2^1)^{-1} * (A_1^0)^{-1} * T = A_n^{n-1} \end{aligned} \tag{2.4}$$

Siendo la matriz de transformación T:

$$T = \begin{bmatrix} nx & sx & ax & px \\ ny & sy & ay & py \\ nz & sz & az & pz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

El problema Cinemático Inverso puede resultar muy complejo de resolver, aun así para sistemas de computo que cuente con los microprocesadores y herramientas de software mas poderosos. La Ecuación 2.4 es solo el punto de partida para plantear, mediante la comparación de elementos en ambos lado de la igualdad, un sistema de ecuaciones linealmente independiente, el cual puede convertirse en un gran inconveniente, dado que las variables a despejar suelen estar en el argumento de funciones trascendentes. [1]

2.8. Herramientas computacionales

2.8.1. Componentes de un sistema de visión por computador

La visión por computador también denominada visión artificial, es una rama de la informática en la que se utilizan ordenadores para el análisis a alto nivel de imágenes digitales [19]. Puede definirse como el proceso de extracción de información del mundo físico a partir de imágenes capturadas y transferidas a un computador. Las tareas que un sistema de visión artificial puede realizar van desde la simple detección de objetos sencillos en la imagen, hasta la interpretación tridimensional de escenas complicadas. El procesamiento digital de imágenes se presenta como una piedra angular en el proceso de toma de decisiones llevado a cabo en las unidades de control de los sistemas, por ejemplo en el área de la robótica, donde muchas veces se cuenta con elementos de captación de imágenes como parte de la instrumentación [20].

La visión artificial esta compuesta por una serie de procesos de obtención, caracterización e interpretación de información de imágenes tomadas de un mundo tridimensional. Estos procesos pueden ser divididos en seis áreas principales:

Captación: proceso a través del cual se obtiene una imagen en dos dimensiones. La imagen es una matriz de $M \times N$ píxeles. Entre los conceptos básicos para el proceso de captación se tienen:

- Profundidad del píxel: define el numero de bits con el que es representado cada píxel.
- Resolución espacial: define las dimensiones mínimas que pueden ser observadas en la imagen digital.

Entre píxeles se pueden dar las relaciones de *vecindad, adyacencia, conectividad y distancia.* Se define como *vecindad* al conjunto de píxeles que se encuentran alrededor de un píxel central, tal como se muestra en la Figura 2.28. La *adyacencia* se utiliza para establecer las fronteras de los objetos y las regiones presentes en una imagen, y la condición para considerar dos píxeles p y q adyacentes es que sean considerados vecinos, y que posean niveles dentro de un

rango definido R. En cuanto a la *conectividad* se dice que si S es un subconjunto de píxeles de una imagen, dos píxeles p y q están conectados en S, si existe un camino entre ellos conformados por píxeles de S. Si R es un conjunto de píxeles de una imagen, constituirá una Región si todos sus píxeles están conectados. Se llama frontera o contorno de R a los píxeles que tienen por lo menos un vecino que no pertenece a R. La distancia $D(p,q)$ es una medida de distancia entre los píxeles $p(x,y)$ y $q(s,t)$ y se define como:

$$D_e(p,q) = \sqrt{(x-s)^2 + (y-t)^2}$$

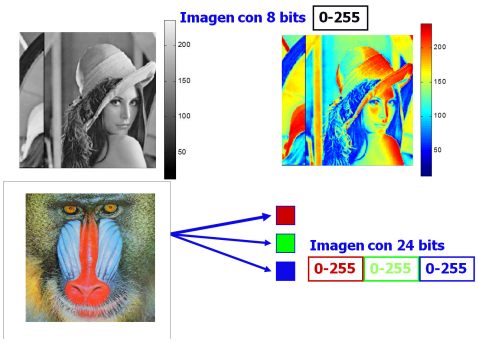


Figura 2.27: Características de las imágenes digitales
Fuente: Wilmer Sanz (2003) [21]

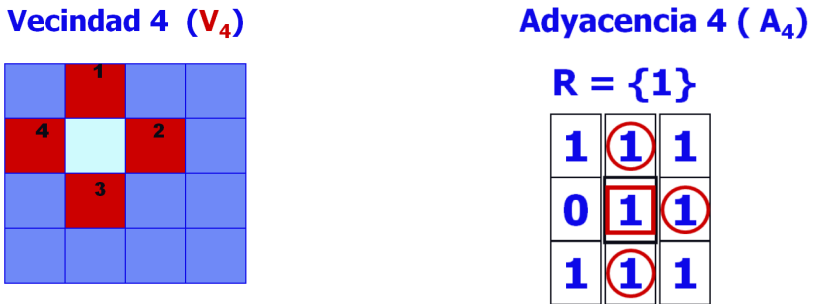


Figura 2.28: Vecindad y adyacencia entre píxeles
Fuente: Wilmer Sanz (2003) [21]

Pre-procesamiento: incluye técnicas tales como la reducción de ruido y realce de detalles. El realce de imágenes es la modificación de la apariencia de una imagen con el objetivo de que la información contenida en ella pueda ser mejor interpretada visualmente respecto a su uso específico. Para tal fin, se realizan operaciones aritméticas y lógicas entre dos imagenes, a partir de las cuales se

genera una nueva imagen con información combinada de ellas. Las operaciones se hacen con píxeles en la misma posición. Es necesario reescalar el resultado de las operaciones para mantener el rango de los niveles digitales. Las aritméticas se definen como suma, resta, multiplicación y división, y las operaciones lógicas se aplican en imágenes binarias, AND , OR, XOR y Complemento. En la Figura 2.29 se muestran algunos resultados de la utilización de estas.

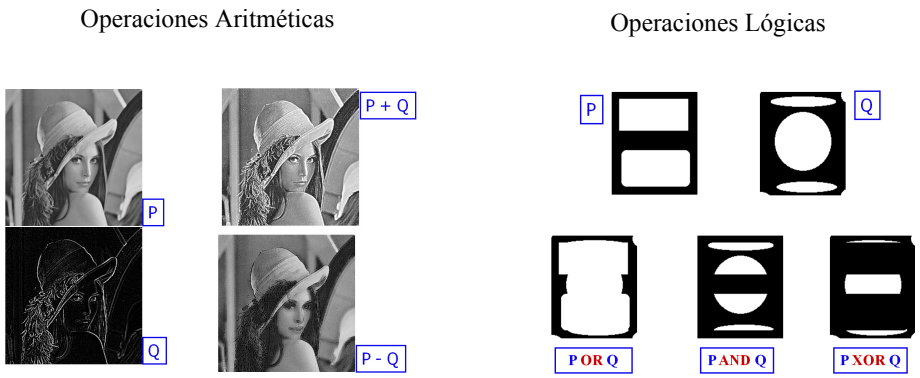


Figura 2.29: Operaciones Aritméticas y Lógicas
Fuente: Wilmer Sanz (2003)[21]

Segmentación: Proceso que divide a una escena en sus partes constituyentes. Entre los principios básicos del proceso de segmentación se distinguen la discontinuidad (procedimientos basados en la detección de bordes) y similitud (procedimientos basados en el crecimiento de regiones). La Segmentación se fundamenta en un conjunto de herramientas matemáticas usadas en el estudio de formas y estructuras (Morfología) de los objetos, los cuales se caracterizan a través de su aspecto geométrico, denominados elementos estructurantes. Estos tienen una forma geométrica simple (disco, cuadrado, línea, etc,) con la cual se compara cada objeto de la imagen, permitiendo discriminar aquellos que poseen propiedades geométricas comunes. Presentan un origen que denominaremos centro del elemento estructurante. Debe elegirse de manera que sus características geométricas (forma, tamaño, orientación, etc) se identifiquen con las características morfológicas de los objetos presentes en la imagen.

Las operaciones morfológicas constituyen un elemento importante en el proceso de segmentación, entre estas operaciones se encuentran la dilatación, erosión, apertura y cierre [22].

Descripción: proceso para obtener características que diferencien un tipo de objeto de otro, por ejemplo tamaño, forma o color.

Reconocimiento: proceso que identifica dichos objetos.

Interpretación: asocia un significado a un conjunto de objetos reconocidos.

Estas áreas se pueden agrupar de acuerdo a la complicación y delicadeza que lleva asociada su implementación, en tres niveles de procesamiento: visión de bajo, medio y alto nivel.

- Visión de bajo nivel: comprende las etapas de *captación* y *pre-procesamiento.* En esta etapa se reducen enormemente la cantidad de datos a tratar. Alguna de las principales funciones a realizar son: filtrado (reducción o eliminación de ruido en una imagen), restauración (perfeccionar una imagen), realce (aumento de contraste de los niveles de gris), extracción de contornos (reduce los datos que contiene la imagen conservando la información de mayor interés), etc.
- Visión a nivel intermedio: comprende las etapas de *segmentación, descripción y reconocimiento.* Estos procesos extraen, caracterizan y etiquetan componentes de la imagen de la visión de bajo nivel. Una de las funciones es el análisis de escenas, reconocimiento de colores o de formas.
- Visión de alto nivel: comprende la etapa de *interpretación.* Se refiere a los procesamientos que tratan de emular la cognición (pertenece al campo de la Inteligencia Artificial y se aplica hoy en día en entornos reducidos y conocidos). Una de las funciones que se realizan es la del reconocimiento de objetos o clasificación por comparación de las características extraídas de la imagen en etapas anteriores [23].

2.8.2. Procesamiento de Imágenes utilizando Matlab

Matlab (abreviatura de MATrix LABoratory, "laboratorio de matrices") es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas de Linux, Windows y Mac. Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes. Además, se pueden ampliar las capacidades de Matlab con las cajas de herramientas (toolbox). Para el caso de manipulación de imágenes se emplea el "*Image Processing Toolbox*".

El procesamiento digital de imágenes se efectúa dividiendo la imagen en un arreglo rectangular de elementos. Cada elemento de la imagen así dividida se conoce con el nombre de píxel. El siguiente paso es asignar un valor numérico a la luminosidad promedio de cada píxel. Así, los valores de la luminosidad de cada píxel, con sus coordenadas que indican su posición, definen completamente la imagen. Todos estos números se almacenan en la memoria de una computadora.

El tercer paso es alterar los valores de la luminosidad de los píxeles mediante las operaciones o transformaciones matemáticas necesarias, a fin de hacer que resalten los detalles de la imagen que sean convenientes. El paso final es pasar la representación de estos píxeles a un monitor con el fin de mostrar la imagen procesada. Existen 3 tipos principales de imágenes:

Imagen de intensidad: es una matriz de datos cuyos valores han sido escalados para que representen intensidades de una escala de grises. Cuando los elementos de una imagen de intensidad son de clase uint8 (enteros almacenados en 8 bits) o de clase uint16 (enteros almacenados en 16 bits), pueden almacenar, respectivamente, 2^8=256 valores en el rango [0-255] o 2^{16}=65536 valores en el rango [0-65535]. Si la imagen es de clase double, los valores son números en punto flotante (que se almacenan en 32 bits).

Imagen binaria (Binary image): imagen que contiene sólo píxeles de color blanco y negro. En Matlab, una imagen binaria es representada por una matriz de tipo logical que contiene 0's y 1's (los cuales representan negro y blanco respectivamente).

Imagen RGB (RGB image): una imagen cuyos píxeles son especificados por 3 valores, uno para cada componente de color (rojo, verde y azul) de cada píxel. En Matlab, una imagen RGB es representada por un array $m \times n \times 3$ de clase uint8, uint16, o double.

2.8.2.1. Algunas funciones importantes de Matlab

Entre los comandos y funciones a utilizar del toolbox de Matlab para la adquisición de la imagen, el procesamiento de la misma, la discriminación de colores (rojo verde o azul), la búsqueda del punto central del objeto y su punto mas cercano y otros, se encuentran las siguientes:

- Para realizar la adquisición de imágenes se debe establecer la conexión entre la cámara web y Matlab. El comando **`vid=videoinput(adaptorname)`** crea un objeto de entrada de video, este representa la conexión entre Matlab y un dispositivo de adquisición de imágenes en particular. Como parámetro posee el *adpatorname*, que es una cadena de texto que especifica el nombre del adaptador utilizado para comunicarse con el dispositivo, en este caso vendrá dado por la cámara web conectada al computador. Se utilizó la función **`imaqhwinfo`** para conocer la información acerca del dispositivo conectado.
- **`C=getdata(vid):`** retorna la información de n cuadros en C, asociada con el objeto de entrada de vídeo *vid.*
- **`imwrite(C,filename,fmt):`** escribe la imagen A al archivo especificado por *filename,* en el formato especificado por *fmt.*
- **`A=imread(filename, fmt)`**: lee una imagen a color o en escala de grises desde el archivo especificado por la cadena de caracteres *filename,* en el formato *fmt.* Para esta aplicación A es una imagen de color RGB, y se representa por tres matrices de dos dimensiones, correspondientes a los planos R, G y B. Para obtener los planos RGB se ejecutan los comandos de la siguiente forma: Para R: $A(:,:,1)$; para G: $A(:,:,2)$; para B: $A(:,:,3)$
- Para visualizar una imagen se utiliza el comando **`imshow(Imagen)`**.
- **`im2bw(A)`**: Crea una imagen binaria a partir de una imagen de una intensidad indexada o RGB, basado en umbralización de la luminancia.
- El comando **`size(A)`** se utiliza para conocer la cantidad de píxeles en los ejes X e Y que tiene la imagen A
- La función **`imfill(A,'holes'),`** rellena los agujeros en una imagen binaria A. Un agujero es un conjunto de píxeles de fondo que no pueden ser alcanzados rellenando el fondo del borde la imagen.
- **`im2= imcomplement (A):`** calcula el complemento de la imagen A. A puede ser una imagen en escala de grises, binaria, o RGB . *im2* tiene la misma clase y tamaño que A.
- **`[X, Y, P]=impixel:`** es una función que habilita la opción de dar click en una determinada imagen, donde X e Y son vectores de igual dimensión los cuales especifican las coordenadas espaciales de los píxeles seleccionados, cuyos valores RGB son retornados en P.

Para la resolución del sistema formado por las ecuaciones (5.13) y (5.14) planteado en el estudio cinemático inverso, se utilizó la función ***fsolve*** del *Optimation Toolbox* de Matlab.

- **[sol, fval]=fsolve(myfun,guess):** esta función resuelve sistemas de ecuaciones no lineales.
- **[F]=myfun(X):** es una función con parámetro de entrada x, arreglo que representa las incógnitas del sistema, y como parámetro de salida F, que es el arreglo donde están escritas las ecuaciones, una en cada fila de F y con la expresión resultante de ser igualadas a 0. De esta manera ***fsolve*** retorna el valor de las incógnitas de *myfun* en *el* vector *sol.*

Cuando se trabaja con la manipulación de datos y se quieren realizar conversiones y arreglos, las funciones mas destacadas a utilizar en esta aplicación son:

- **dec2bin(x):** esta función convierte un valor decimal en uno binario de tipo string.
- **num2str(valor)**: convierte un valor decimal en uno tipo string.
- **strcat[x1...xn]:** es una función que concatena valores string horizontalmente.
- **base2dec('strn', base)**: esta función convierte una dato de ***base*** N string, en uno decimal.

En cuanto al manejo de librerías Matlab ofrece una gran cantidad de funciones que facilitan su uso, entre las mas utilizadas se encuentran:

- **p = libpointer('type',value):** esta función devuelve un puntero del tipo de datos especificado e inicializado con el valor suministrado.
- **calllib ('libname', 'funcname', arg1, ..., argN):** esta función se encarga de llamar a la función *funcname* en *libname* libreria, pasando argumentos de entrada arg1 hasta argN.
- **get (h, 'Value'):** la función devuelve el valor del objeto gráfico identificado por h [24].

2.8.2.2. Archivos MEX para la utilización de la librería HIDAPI con Matlab

Es posible llamar desde Matlab funciones programadas en C, C++ y en Fortran como si fuesen funciones propias de Matlab. Los programas en C y Fortran que se pueden llamar desde Matlab son las MEX-files o archivos MEX. Los MEX-files son subrutinas ligadas dinámicamente que el interpretador de Matlab puede automáticamente cargar y ejecutar. MEX proviene de las palabras "Matlab EXecutable". Las funciones MEX tienen varias aplicaciones:

- Evitan tener que rescribir en Matlab funciones que ya han sido escritas en C o Fortran.
- Por motivos de eficiencia puede ser interesante rescribir en C o Fortran las funciones críticas de una aplicación determinada.

En particular, se necesita un compilador soportado por Matlab y es necesario tener el paquete SDK (Software Development Kit) de Microsoft correspondiente. Se puede buscar la lista de compiladores soportados y compatibles en la página web de Mathworks Inc., que dependen también de que versión de Matlab se esté usando, en este caso la R2012a. Si se tiene hecho un programa independiente en C, se debe usar una función main(). Matlab se comunica con la MEX-file usando una gateway routine (rutina de puerta de entrada). Por otro lado, la función de Matlab que crea una rutina se llama mexfunction. Matlab guarda los arreglos en una variable de tipo mxArray. Se utiliza mxArray en el programa en C, para pasar datos de Matlab hacia y desde la MEX-file [25].

2.8.3. Interfaz Gráfica

Se puede definir la interfaz gráfica como un artefacto interactivo, que por su diseño y a través de ciertas interfaces humanas, posibilita la interacción de una persona con el sistema informático, haciendo uso de las gramáticas visuales y verbales (signos gráficos como iconos, botones, menús y verbales como tipografía). Como todo artefacto exige por parte de la persona que interacciona las capacidades fisiológico-cognitivas mínimas, para poder interpretar adecuadamente los signos, y poder realizar acciones efectivas sobre la propia interfaz.

2.8.3.1. Entorno de desarrollo GUI

Matlab GUI es el entorno de programación gráfica (ver Figura 2.30 (a)) que hace posible el desarrollo de aplicaciones que permiten una interacción eficiente entre el usuario y el ordenador. Las GUI (también conocidas como interfaces gráficas de usuario) permiten un control sencillo de las aplicaciones de software.

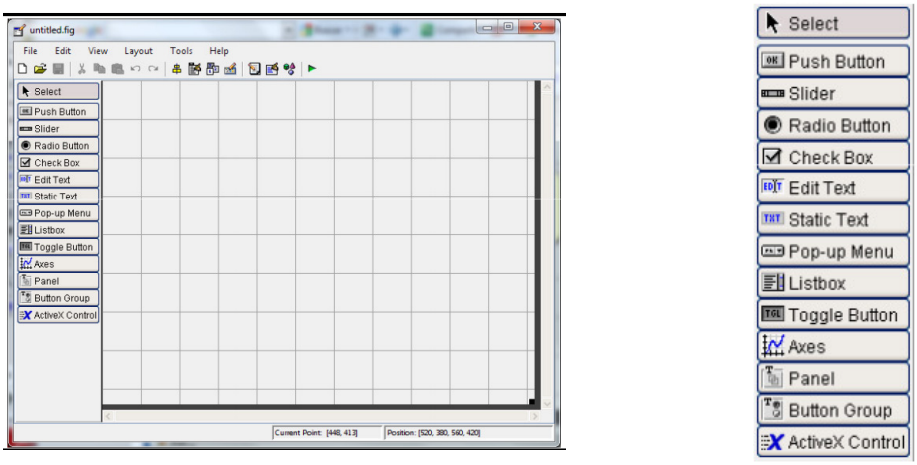


Figura 2.30: a)Entorno de diseño. b)Componentes
Fuente: Diego Barragán (2008)[26]

El editor permite construir interfaces arrastrando y soltando componentes en el entorno de diseño de la GUI. Todas las GUIs creadas con guide empiezan con una función inicial (callback) que se invoca cuando se invoca la interfaz Matlab. La operación automática de guardado (save) genera un fichero *.m* y un fichero *.fig* . El fichero *.fig* contiene el diseño del GUI en binario y el fichero *.m* contiene el código que controla el GUI [26].

2.8.3.2. Componentes dentro de GUI

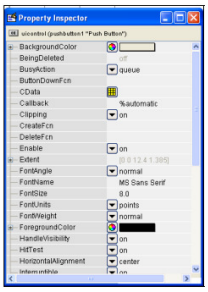
Por lo general, la GUI incluye controles tales como menús, barras de herramientas, cuadros de diálogo, controles de interfaz de usuario (como botones y controles deslizantes) y contenedores (como paneles y grupos de botones). GUIDE (entorno de desarrollo de GUI) proporciona herramientas para diseñar interfaces de usuario y genera de manera automática el código de Matlab para construir la interfaz, el cual se puede modificar para programar el comportamiento de la aplicación (ver Figura 2.30 (b)). A fin de ejercer un mayor control

sobre el diseño y el desarrollo, también se puede crear código de Matlab que defina las propiedades y los comportamientos de tod

os los componentes [24].

Control	*Valor de estilo*	*Descripción*
Check box	'checkbox'	Indica el estado de una opción o atributo
Editable Text	'edit'	Caja para editar texto
Pop-up menu	'popupmenu'	Provee una lista de opciones
List Box	'listbox'	Muestra una lista deslizable
Push Button	'pushbutton'	Invoca un evento inmediatamente
Radio Button	'radio'	Indica una opción que puede ser seleccionada
Toggle Button	'togglebutton'	Solo dos estados, "on" o "off"
Slider	'slider'	Usado para representar un rango de valores
Static Text	'text'	Muestra un string de texto en una caja
Panel button		Agrupa botones como un grupo
Button Group		Permite exclusividad de selección con los radio button

a)



b)

Figura 2.31: a) Caracteristicas de los componentes b) Entorno de Property Inspector
Fuente: Diego Barragán (2008) [26]

En cuanto a las propiedades de los componentes es posible acceder a estas a través de la opción Property Inspector, la cual permite personalizar cada elemento. Al hacer click derecho en el elemento ubicado en el entorno, una de las opciones mas importantes es *View Callbacks*, la cual al ejecutarla, abre el archivo *.m* asociado al diseño y posiciona el cursor en la parte del programa que corresponde a la subrutina que se ejecutara cuando se realice una determinada acción sobre el elemento que se esté editando. Por ejemplo, al ejecutar *View Callbacks* en un elemento tipo *Push Button*, la parte del programa a donde lleva es de la siguiente manera:

```
function pushbutton1Callback(hObject,eventdata,handles)
```

2.8.3.3. Funcionamiento y manejo de datos entre los elementos de una aplicación y el archivo .m

Una aplicación GUI consta de dos archivos *.m* y *.fig*. El archivo *.m* es el ejecutable y el *.fig* la parte gráfica.

Todos los valores de las propiedades de los elementos (color, valor, posición, string...) y los valores de las variables transitorias del programa se almacenan en una estructura, los cuales son accedidos mediante un único y mismo puntero para todos estos. Para la asignación del puntero:

```
handles.output=hObject
```

Donde `handles`, es el puntero a los datos de la aplicación. Esta definición de puntero es salvado con la siguiente instrucción:

`guidata(hObject,handles):`

`guidata`, es la sentencia para salvar los datos de la aplicación. Esta guarda las variables y propiedades de los elementos en la estructura de datos de la aplicacion, por lo tanto, como regla general, en cada subrutina se debe escribir en la ultima linea, de esta forma se garantiza que cualquier cambio o asignación de propiedades o variables quede almacenado.

2.8.3.4. Sentencias get y set

La asignación u obtención de valores de los componentes se realiza mediante las sentencias set y get respectivamente. La estructura es la siguiente:

`variable=get(handles.nombredelcomponente, 'Value');` Obtener el valor del componente y asignarlo a *`variable`.*

`set(handles.nombredelcomponente, 'String',variable)` Asignar el valor *`variable`* al componente [26].

2.8.4. Roboworks

Roboworks es un modelador tridimensional para la simulación de elementos mecánicos como se muestra en la figura. La mayoría de los software de análisis y visualización tales como Matlab, Mathcad y LabVIEW, soportan gráficos limitados a tablas y gráficos. Roboworks añade capacidades de modelado y animación 3D en estos paquetes de software. También permite hacer análisis en el software de preferencia y visualizar y animar los resultados. Adicionalmente soporta animación interactiva a través del teclado (animar a medida que se hace el modelado), a través de un archivo de datos, o por medio de RoboTalkTM.

El usuario de Roboworks tiene al alcance las herramientas necesarias para realizar un modelo en 3D y animarlo a través de un archivo .dat creado por el mismo o por medio de archivos ejecutables creados en Matlab, C++, LabView, etc. [27]. Para la realización de este proyecto Roboworks es utilizado para evaluar la veracidad de los modelos cinemáticos directo e inverso, permitiendo simular los movimientos del Rhino XR-3 desde un archivo generado por Matlab a través de la Interfaz Gráfica.

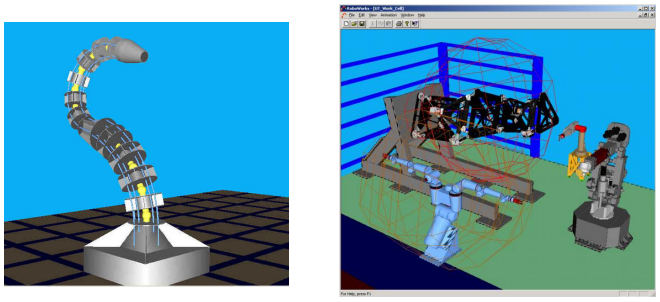


Figura 2.32: Ejemplos de modelos tridimensionales diseñados en Roboworks
Fuente: Newtonium-Roboworks[27]

3 Marco Metodológico

3.1. Tipo de la investigación

En esta fase de la investigación se define la estrategia para el tratamiento metodológico categorizando la investigación en la modalidad de proyecto factible, ya que este se define como: la investigación, elaboración y desarrollo de una propuesta de un modelo operativo viable, cuyo propósito es la búsqueda de la solución de un problema [28].

El modelo que se implementará para ejecutar la investigación, consistirá en efectuar un diseño viable para realizar un control de variables específicas a partir de una base diagnóstica, por esto, éste trabajo se encuentra enmarcado en el diseño de un modelo factible. La propuesta debe tener apoyo en una investigación de tipo documental, de campo o un diseño que incluya ambas modalidades.

3.2. Diseño de la investigación

La presente investigación es de tipo factible, debido a que constituye un proceso sistemático, riguroso y racional de la recolección, tratamiento, análisis y presentación de datos basado en una estrategia de recolección directa de la realidad de la información necesaria para la investigación. Tiene apoyo en una investigación de tipo documental, siguiendo la descripción de Alfonso (1995) quien la define como un procedimiento científico, un proceso sistemático de indagación, recolección, organización, análisis e interpretación de información o datos entorno a un determinado tema[29].

Existen una serie de procedimientos para desarrollar la investigación documental que compone el proyecto factible, los cuales se mantendrán presente para el desarrollo del proyecto, haciendo de éste un proceso más eficiente. Los pasos

a seguir serán los siguientes: selección y delimitación del tema, recolección de información, análisis de los datos, redacción de la investigación y presentación final.

3.3. Técnicas e instrumentos de recolección de datos

Se utilizan una variedad de métodos a fin de recopilar información sobre una situación existente como la revisión de registros, revisión de documental y observación. Cada uno posee ventajas y desventajas, pero utilizarlos de manera conjunta ayuda a asegurar una investigación completa.

Revisión Documental: se refiere a la consulta de libros de texto, artículos, revistas científicas, publicaciones, tesis de grado y a los antecedentes que serán base para el análisis en el Trabajo Especial de Grado. Se realiza la revisión documental definiendo cual es el material de interés, y luego se procede a una clasificación del material con el objeto de ubicarlo en una de las partes del proyecto. De esta forma se podrá conocer al detalle el funcionamiento dinámico del brazo robot Rhino XR-3, y se lograrán elaborar esquemas de trabajo basados en investigaciones afines.

Revisión de registros: consiste en la inspección de información de tipo técnico, tales como: planos, diagramas, manuales, reportes técnicos y recursos bibliográficos con relación al objeto de estudio.

Observación: radica en establecer una relación con el objeto de estudio, permitiendo generar una teoría y un método adecuado para que la investigación se oriente de manera correcta, y el trabajo de campo arroje datos exactos y confiables. Puede realizarse tanto de forma directa como indirecta. Directa al observar y recopilar información dentro del conjunto de elementos que se quiere conocer e investigar, y de manera indirecta mediante la utilización de instrumentos que muestren la problemática subjetivamente, produciendo una mayor proximidad a la realidad.

3.4. Etapas de la investigación

Esta investigación se realizará cumpliendo paso a paso los objetivos a través de las siguientes etapas:

Fase I: Diseño del control de motores del brazo robot Rhino XR-3. En esta fase se determinarán las características de la herramienta de trabajo del Rhino XR-3. Se diseñarán circuitos prototipos para controlar los movimientos de las articulaciones del brazo robot. Posteriormente se realizarán las simulaciones necesarias para la corrección de errores y lograr el perfecto funcionamiento del mismo. Luego se procederá a la construcción de circuitos de control sobre el XR-3, donde se ensayarán reiteradamente las alternativas hasta conseguir una solución satisfactoria que dé cumplimiento al control preciso de los motores.

Fase II: Construcción de tarjeta de control para el brazo robot. Se diseñará y construirá un nuevo circuito impreso de la Unidad de Control para el RHINO XR-3.

Fase III: Comunicación USB circuito-computador. Se adaptará el sistema de comunicación del robot a USB, y así pueda ser manejado desde cualquier computador con las características listadas en las limitaciones de la investigación. Se utilizará un fichero estándar que suministra Microchip para hacer compatible los PIC‘s con el puerto USB; para el envío y recepción en el PC a 64 bytes de datos por milisegundo. También se empleará el software de simulación Proteus 7.4 para validar el funcionamiento del hardware que mide y transmite la variable física por medio de USB.

Fase IV: Determinar modelo cinemático inverso del brazo robot Rhino XR-3. Se hará el estudio de la cinemática directa del brazo robot XR-3. Seguido de esto se harán los cálculos necesarios para plantear un modelo que resuelva la cinemática inversa que pueda ser ejecutada desde Matlab, y así obtener los ángulos que debe moverse cada articulación del robot para realizar determinada tarea.

Fase V: Evaluar el modelo cinemático inverso planteado mediante un software especializado. Haciendo uso de Matlab se realizaran simulaciones que permitan comprobar el correcto funcionamiento de la cinemática inversa mediante graficas 2D y 3D. También se diseñará en el modelador tridimensional RoboWorks un modelo a escala que muestre el funcionamiento del robot en el entorno en el que trabajará utilizando un archivo de datos .dat creado a través de una Interfaz Gráfica.

Fase VI: Detección de objetos mediante cámara web. En esta fase, lo primero es realizar la calibración de la cámara, luego se harán las conversiones de píxel-distancia requeridas para el plano de trabajo que la imagen capturará. Seguido de esto el procesamiento de la imagen hace posible la discriminación de objetos de acuerdo a colores (rojo, verde, azul). Se obtendrá la ubicación de un píxel central de una región en la fotografía, y de esta manera estimar la ubicación espacial del objeto. Además en esta fase se procesará la imagen para conocer la orientación de objetos de forma cuadrada y rectangular, logrando la orientación adecuada de la herramienta de trabajo del robot.

Fase VII: Diseño de Interfaz Gráfica que permita controlar al Rhino XR-3. Se diseñará una interfaz en el entorno de MATLAB que permita al usuario relacionarse con el robot, y posicionar lo de tres formas: la primera consiste en que el usuario introduzca las coordenadas donde quiere que se posicione el brazo robot $\overrightarrow{P}(x, y, z, \theta_5, Gripper)$ o que $P(x, y)$ sean obtenidas mediante clicks sobre la imagen, la segunda forma es que las coordenadas de un objeto en particular puedan ser obtenidas por medio del procesamiento de una imagen del plano de trabajo y dado un color especifico tome el objeto y lo lleve a un punto definido por el usuario $Pf(x, y, z)$, y la tercera y última forma consiste en que el usuario proporcione directamente los valores de $\overrightarrow{\theta}_1$, $\overrightarrow{\theta}_2$, $\overrightarrow{\theta}_3$, $\overrightarrow{\theta}_4$, $\overrightarrow{\theta}_5$, $\overrightarrow{G}$. Una vez obtenidos los valores de los ángulos de cualquiera de las maneras anteriores, el usuario tendrá la opción de simular los movimientos con Roboworks, y hacer la comunicación USB.

4 Diseño del hardware programable

En este capítulo se expone la descripción del circuito que constituye la Unidad de Control Cinemático (UCC) del robot XR-3, así como las consideraciones más importantes de diseño y experiencias durante su desarrollo. El circuito diseñado tiene por objetivos controlar cada uno de los motores para lograr una posición angular deseada, y comunicar el circuito por USB con un computador que le emita órdenes de posición específicos.

4.1. Consideraciones de diseño

4.1.1. Elección del microcontrolador

El primer problema de diseño se presenta al adaptar los requerimientos del circuito a la disponibilidad de microcontroladores accesibles en el mercado que incluyan características específicas pertinentes a este proyecto como lo son: Hardware USB, PWM y ADC presentados en la Sección 2.4.5 . Se analizó únicamente el fabricante Microchip por ser empleado para la docencia en la Universidad de Carabobo.

La unidad de control cinemático debe controlar seis motores: cinco en las articulaciones y uno para el actuador. Cada motor requiere al menos seis (6) pines dedicados para su control: tres para el sentido de giro y habilitación (puente H), dos para sensores de posición y velocidad (encoders), y uno para la calibración (sensor de fin de carrera); resultado un total de 36 entradas o salidas (I/O) necesarias para el manejo de todas las variables, resumidas en la Figura 4.1 (a). Para simplificar el diseño es posible reducir el número de I/O prescindiendo de uno de los encoder y el pin de habilitación reduciéndose el total a 24 I/O requeridos. Adicionalmente a las I/O antes mencionadas, el

número de pines que debe tener el circuito integrado debe incluir además los necesarios para la comunicación USB, para alimentación y botones de reset.

Los microcontroladores de Microchip® pueden poseer 8, 14, 20, 28 y 40 pines en total, siendo los de la familia PIC18 de 28 o 40 pines los únicos que incorporan el circuito de alta velocidad que permite la comunicación USB. Esta cantidad de pines resulta insuficiente para el manejo de todas las variables requeridas. Se exploraron diversos diseños que empleen más de un microcontrolador, con uno en respaldo serial o bien con microcontroladores más pequeños dedicados a cada motor guiados por un maestro que use el puerto USB, sin embargo los circuitos resultaron engorrosos y complejos.

El diseño final se valió de la multiplicación de las variables de cada motor. Ello permitió utilizar el dispositivo integrado PIC18F2550, el cual es relativamente fácil de encontrar en el mercado, posee un puerto USB y las salidas suficientes para la manipulación multiplexada de las variables requeridas e inclusive el empleo de una pantalla LCD para verificar los datos y procesos. El diseño desarrollado en este trabajo limita a la unidad de control a mover un solo motor a la vez, sin embargo, hace el hardware compacto, económico y más simple, presentando muchas ventajas sobre las unidades de control precedentes. La distribución multiplexada de las I/O de motores y del microcontrolador son expuestos en las Figura 4.1 (b) y (c) respectivamente.

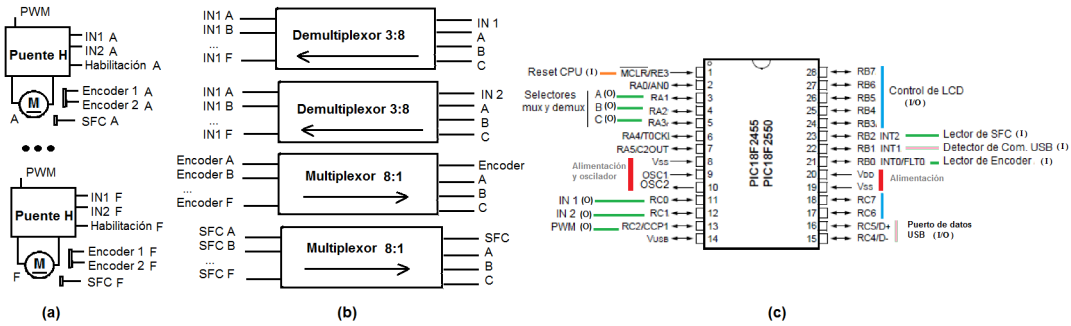


Figura 4.1: Esquema general del diseño. (a) Control de motores (b) Multiplexación (c) I/O del microcontrolador

4.1.2. Actuación sobre motores

Las diferentes posiciones del robot cambian la geometría y por tanto la inercia del mismo. Esto hace que el consumo de potencia de cada motor dependa de cada configuración de ángulos dada. Por esta razón, es desconocida la cantidad

de energía requerida para mover la estructura del robot. Además, el fabricante solo indica los limites de corriente soportada por los motores, más no especifica los valores de corriente en régimen permanente (especificaciones técnicas Sección 2.2.4), por lo que solo las pruebas realizadas permitirán estimar la potencia que deben manejar los componentes que actuarán sobre los motores.

Durante el desarrollo de este trabajo se probó la implementación independiente de todos y cada uno de los controladores electrónicos señalados en la Sección 2.3.4, obteniendo los mejores resultados empleando el L6203B, por manejar la potencia requerida por los motores y por su robustez en la implementación física luego de numerosas pruebas. A continuación se resumen las experiencias y consideraciones más destacadas.

Se exploró y descartó un diseño que emplea los triac GDA3AT accionados por optoacoplador, y comprende un único puente H transistorizado para la alimentación de un ramal principal común a todos los motores, donde un Triac independiente para cada ramal habilita la energización del motor que se desea mover. Este diseño funcional fue implementado como se muestra en la Figura 4.2, requiriendo menor área y costo que el empleo de puentes H integrados. Sin embargo, se obtuvo inestabilidad en la respuesta, además de dificultad en la sincronización de la inversión de giro y el apagado del triac, posiblemente debido a una inadecuada compensación o influencia de la fuente conmutada utilizada. Por tanto descartó el uso de componentes discretos.

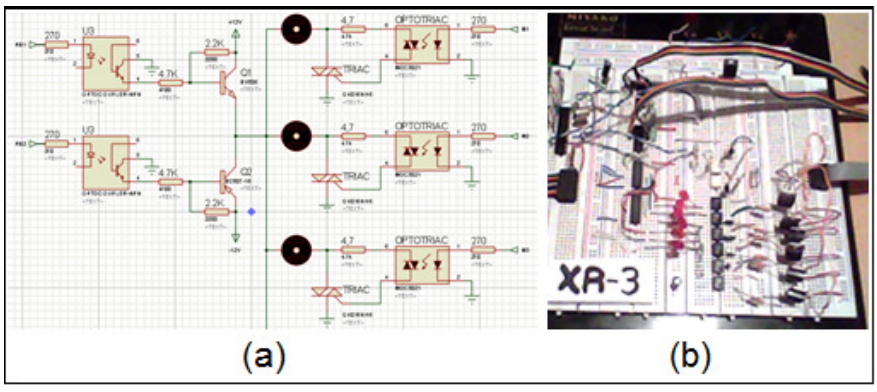


Figura 4.2: Controlador diseñado empleando Triac. (a) Esquemático (b) Implementación.

El circuito integrado L298n, que a diferencia de otros incorpora dos puentes H integrados, no es capaz de soportar la corriente de los motores grandes, sin embargo si es capaz de mover los dos motores pequeños del XR-3, que son la apertura y el giro de la pinza. Por simplicidad de los algoritmos, se procuró un

único diseño para el control de todos los motores, descartando el uso de este componente.

Se logró controlar todos los motores con el integrado puente LMD18200T, rindiendo buenos resultados con sus 3A de carga permisibles y corriente pico de 5A (tal como los motores utilizados), por lo que se procedió a la implementación física de un primer módulo de control de motores. Sin embargo, durante las pruebas de calibración realizadas, resultaron dañados varios de los dispositivos, haciendo que solo se activen en un único sentido de giro. Se sospecha que las corrientes de arranque o las corrientes inversas de los motores produjeron el deterioro del componente.

Para optimizar el circuito, se empleó un componente con mayor capacidad de corriente, el L6203b, 4 A de corriente de carga y hasta 6A de corriente pico, protegido de corrientes inversas con diodos de fuga y capacitores, resultando el circuito de la Figura 4.3 (a), implementado como se indica en la Figura 4.3 (b).

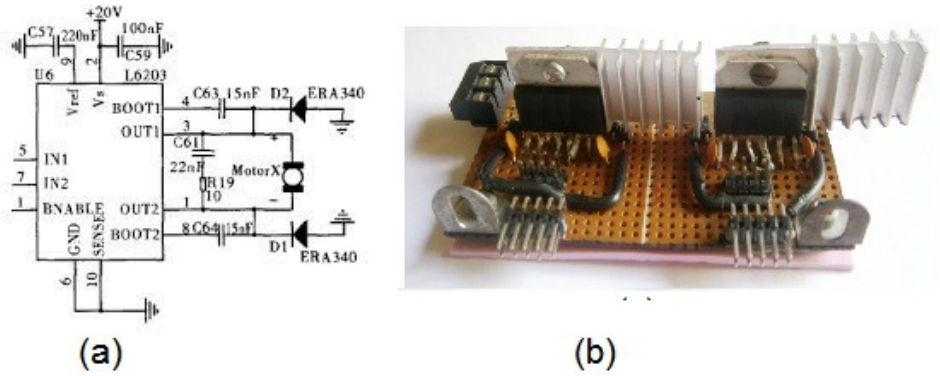


Figura 4.3: Módulo controlador de un motor con el L6203b. (a) Esquemático (b) Implementación.

4.1.3. Corrientes parásitas y acoplamiento

Las pruebas indicaron que las señales de control no tienen total aislamiento sobre la carga, de lo que se infiere la consideración de diseño más importante: el microcontrolador debe ser totalmente aislado de la alimentación, entradas y salidas de los circuitos integrados puente H, ya que no posee un aislamiento completo a pesar de las indicaciones del fabricante, produciendo corrientes inversas que desestabilizan al microcontrolador. Para ello se emplearon optoacopladores como los mencionados en la Sección 2.4.1. Se probó una fuente

de 5V independiente para el microcontrolador y otra de 5V para la multiplexación y el circuito lógico que controla los puentes H, obteniendo únicamente resultados positivos si se aíslan completamente las referencias de «tierra» de los circuitos, de manera tal que el microcontrolador permanezca en «flotante» con respecto al resto del circuito, conectándolo únicamente por medio de los opto-acopladores. Analizando los efectos de referencias independientes tanto para Vcc como Vee, se hace a la carcasa del circuito integrado, susceptible a que un potencial estático afecte la tensión base-emisor produciendo un falso positivo, sin embargo para este diseño se consideró un efecto despreciable en un circuito de baja frecuencia en los bien definidos niveles lógicos TTL a corta distancia.

Para mejorar la robustez de la señal, se desarmó el circuito que constituye el encoder del motor, obteniendo información de su componente fundamental, el comparador diferencial dual LM93A, cuya salida en colector abierto es la generadora de las señales leídas en el Encoder. Por tanto, para mejorar la firmeza de la señal se implementó entre el terminal correspondiente al encoder y la entrada del optoacoplador una resistencia de Pull UP, ver Figura 4.4, que hacen circular por ella la corriente para encendido del LED interno al optoacoplador, y no a través del largo cable hasta el encoder ubicado en el motor. Este diseño se justifica con la caída de tensión que deteriora la señal TTL [9].

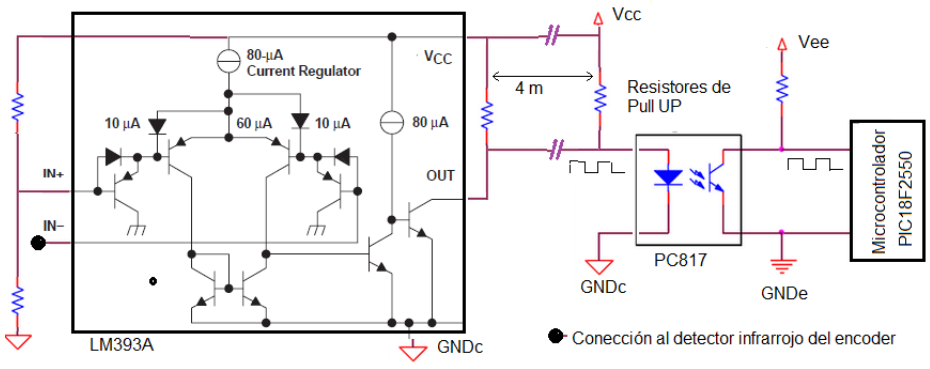


Figura 4.4: Transmisión de señales de los Encoder.

4.1.4. Fuentes de alimentación

Durante las pruebas de control de los motores del XR-3, se determinó que es la tensión y no la corriente el factor determinante en el movimiento de

las articulaciones del robot, debido quizás a la caja de engranes a la que está acoplado el eje, siendo óptimo un voltaje comprendido entre los 15 y 18 voltios. Una fuente de 13.5V (cargador de computador portátil) con capacidad de 1A de corriente fue suficiente para hacer mover todas las articulaciones excepto la del motor D (cadera), que se sospecha responsable de su destrucción por la corriente exigida. También las pruebas permitieron verificar la conveniencia de habilitar el Brown Out Reset del microcontrolador, que lo hace reiniciar como protección (ver Sección 2.4.4).

En el resto de las pruebas preliminares se empleó una fuente de laboratorio con selector de voltaje de salida DC y hasta 5A. Durante el arranque de los motores se activó instantáneamente un led indicador de protección contra sobrecorrientes de la fuente de poder, para lo cual se hace necesario un protocolo PWM y capacitores de amortiguación para el de arranque suave de los motores.

Los experimentos evidenciaron que las fuentes conmutadas de computadores no deben ser usadas para la alimentación de los motores, ya que son susceptibles a daños debido a los cortocircuitos generados en el arranque o parada de los motores. También debe tenerse presente como posible causa de falla que el circuito estará conectado al computador mediante bus USB, el cual provee alimentación en 3V como se indicó en la Sección 2.5.2, y por tanto se compartirán inevitablemente las referencias de «tierra» del computador, el microcontrolador y el circuito conectado a él. Se exploró la posibilidad de un buffer distribuidor USB para aislar las referencias, sin embargo, el aislamiento del microcontrolador al resto del circuito fue suficiente para garantizar la estabilidad de su funcionamiento.

4.1.5. Configuración para la comunicación USB

Para la implementación física de la comunicación USB deben conectarse adecuadamente los terminales del conector USB tal como se indica en la Sección 2.5.2.1, teniendo especial cuidado que el conector hembra (del dispositivo) es la reflexión simétrica del conector macho al que hacen referencia la mayoría de bibliografías de internet. Al conectarse al microcontrolador debe verificarse una tensión muy cercana a los +3V en el pin VUSB (pin 14 en el PIC18F2550), que indica una buena configuración. Resulta útil el empleo de un Bootloader (como el mostrado en la Sección 2.5.6), ya que su funcionamiento desde el computador asegura una adecuada configuración, así como conexión y estado del cable.

Se observó un comportamiento irregular al implementar las «interrupciones por flanco de bajada» del microcontrolador, señaladas en la Sección 2.4.4, en las cuales el hilo de proceso central se interrumpe por un cambio en un puerto específico, quizás debido al solapamiento con la interrupción requerida para mantener activo el bus USB (cada 1ms). Probablemente cerrar la comunicación USB permita sacar más provecho a las bondades del microcontrolador, sin embargo no fue posible realizar la desconexión y reconexión del dispositivo por software.

Es fundamental la selección de la «palabra de configuración del dispositivo» en la ventana Edit Project de MikroC. Para ello debe seguirse el procedimiento resumido específicamente en la Sección 2.5.8, a lo que deberá resultar una ventana con propiedades similares a la mostrada en la Figura 4.5; donde el procesador trabaja a 4 MHz sincronizado con un PLL de 96 MHz usado para el bus USB. Las configuraciones usadas para los descriptores del proyecto empleando el Tool HID Terminal se muestran en la Figura 4.6.

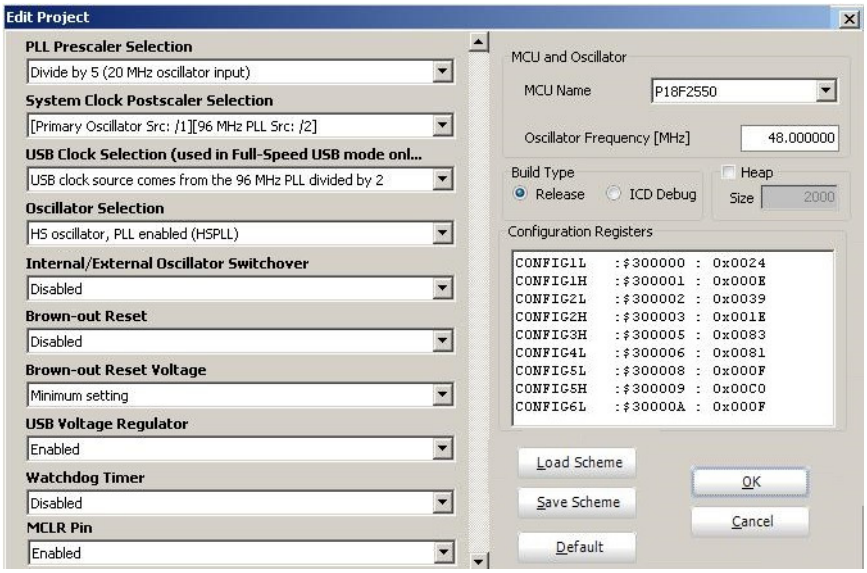


Figura 4.5: Ventana de edición de proyecto en MikroC Pro For PIC.
Fuente: Mikroelectronica- MikroC

Es importante destacar que en algunas versiones de MikroC For PIC, el menú de ayuda hace referencia a las direcciones de memoria específicas para el manejo del bus USB. Si una variable no pertenece a esa región de memoria, la implementación simplemente no funcionará. Es fundamental el conocimiento del hardware ya que el compilador es una herramienta de ayuda limitada.

Otra observación importante a considerar en la programación es referida al ajuste del oscilador, preescalador, Watch Dog Timer u otras características de

Figura 4.6: Ventana de configuración de descriptores apara el dispositivo conectado USB
Fuente: Mikroelectronica- MikroC)

la palabra de configuración. Para ello debe usarse únicamente la herramienta que ofrece MikroC, en caso de acceder a alguno de estos registros en el código fuente (ejemplo OSCON = 0x26, para configurar el oscilador). Entonces aunque la configuración sea correcta al implementar el circuito, no podrá simularse el circuito en Isis Proteus por razones intrínsecas al software.

4.1.6. Diseño modular

La unidad de control consta de seis motores, cada uno con seis variables manipuladas: Encoders (Ea, Eb), sensor de fin de carrera (SFC), controladores del puente H (IN1, IN2) y habilitación (En). Por otra parte tenemos que el control de estas variables independientes será manejado mediante mux y demux, resultando en pines consecutivos de entrada o salida. Esto implica un direccionamiento del cableado muy engorroso para una sola tarjeta de circuito impreso, como ya se mostró en la Figura 4.1. Para disminuir la posibilidad de error, facilitar la reparación y mejoras posteriores del dispositivo, se planteó un diseño subdividido en módulos interconectados, descritos a continuación:

- Módulo de control central: Contiene al microcontrolador, el puerto USB, una pantalla LCD, y salidas opto acopladas y multiplexadas, con conectores agrupados según el tipo de variable manipulada.
- Módulo de distribución: Únicamente posee el cableado de distribución de las variables a cada motor correspondiente, conectando el módulo

de control central con los módulos de control de motores cuyo conector organiza las variables según el motor.

- Módulos para control de motores: Fundamentados por un puente H integrado, posee también las protecciones y el conector que transfieren la señal recibida de los encoder y sensor de final de carrera.

4.2. Descripción del hardware

La Unidad de Control Cinemático es resultado de la acción conjunta de un circuito y un software de computador comunicados por USB. Una vista del controlador y el actuador son mostrados en la Figura 4.7.

Una vez conceptualizada la arquitectura del hardware, resumida en el diagrama de la Figura 4.8, se procedió a probar y optimizar el diseño de conexión a cada componente por separado; bien sea mediante simulaciones en el software Isis Proteus (señalado en la Sección 2.4.7), o mediante implementaciones físicas en el protoboard (o tabla de pruebas), tal como se muestra en la Figura 4.9. A continuación se describen los planos circuitales, diseño de tarjetas de circuito impreso y fotografías de la implementación final de cada módulo elaborado.

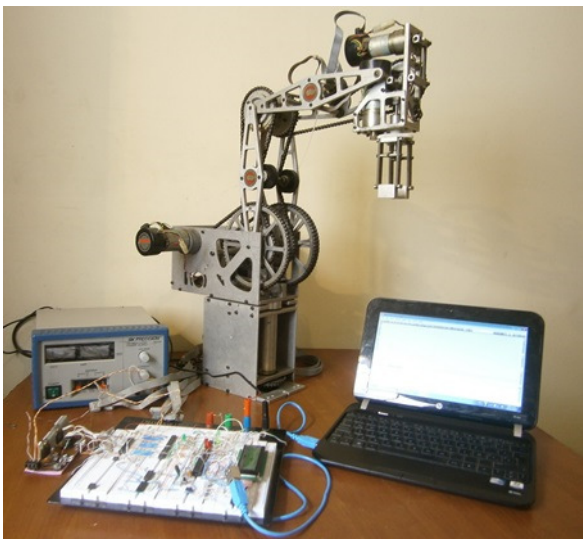


Figura 4.7: Vista de la Unidad de Control Cinemático y todos sus elementos.

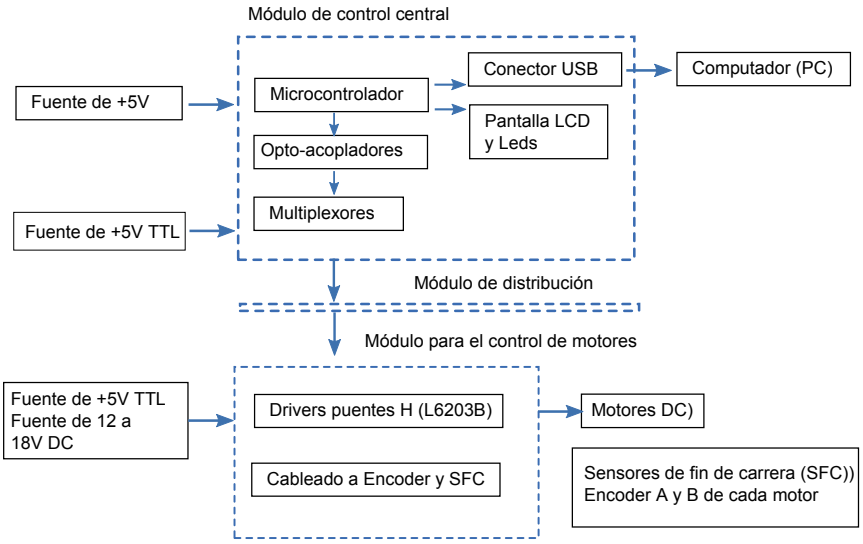


Figura 4.8: Diagrama de bloques asociado al diseño del hardware.

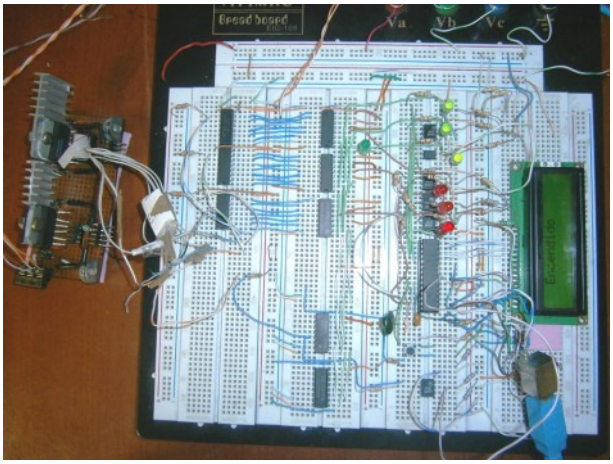


Figura 4.9: Implementación final en protoboard de la Unidad de Control Cinemático para el autómata antropomórfico XR-3

4.2.1. Módulo de control central

El plano modular de este circuito realizado en Isis Proteus se muestra en la Figura 4.10. Las planchas para circuito impreso en escala 1:1 se muestran en la Figura 4.11.

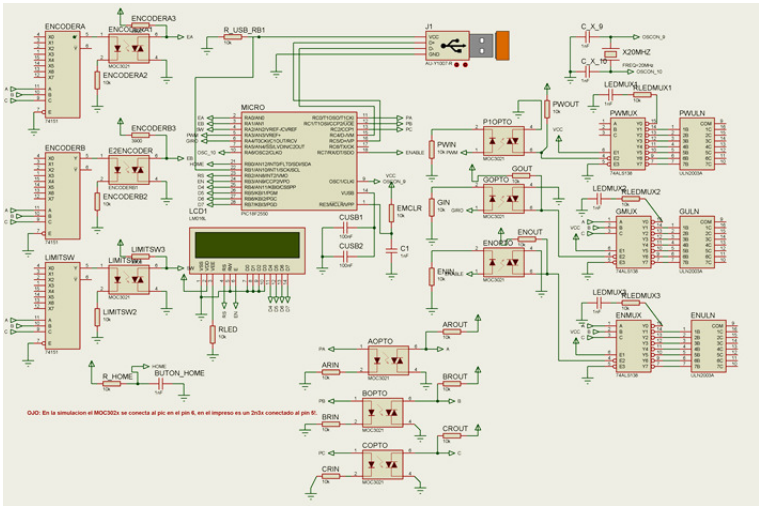


Figura 4.10: Esquemático del circuito del módulo de control central, realizado con el software Isis Proteus.

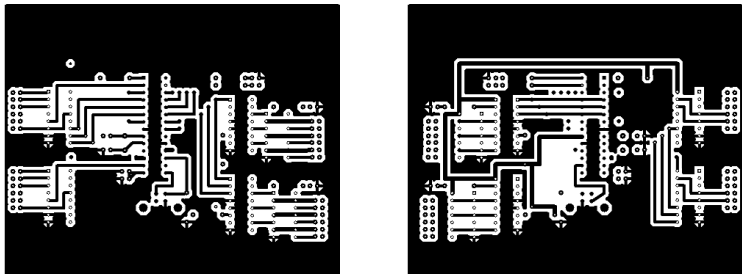


Figura 4.11: Diseño para el circuito impreso, cara superior (a) e inferior (b).

4.2.2. Módulo de distribución de señales

Se encarga de reagrupar las variables de cada motor según su naturaleza, para asignar a un multiplexor o demultiplexor la variable correspondiente al tipo de señal (todos los Encoder, todos los SFC...). Este módulo se implementó con cableado de cobre de tipo telefónico, por ser una alternativa factible al montaje superficial multicapa para distribuir buses de señales entrecruzadas. Fotografías de su implementación se muestran en la Figura 4.12.

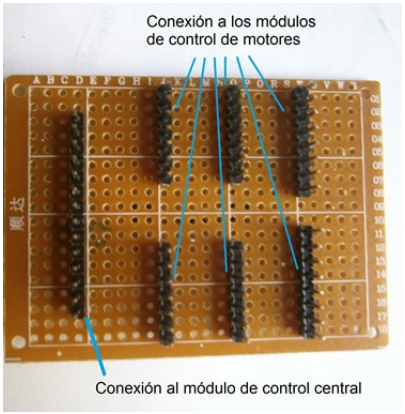


Figura 4.12: Módulo de distribución de señales.

4.2.3. Módulo de control de motores

El circuito que controla los motores fue desarrollado empleando el componente puente H integrado L6203B, como se muestra en la Figura 4.13. Este incorpora la amortiguación de picos de corriente provenientes del motor durante el arranque o inversiones bruscas de giro o exceso de par aplicado al eje. Para una implementación más robusta se sugiere emplear la tarjeta de circuito impreso diseñada que se muestra en la Figura 4.14.

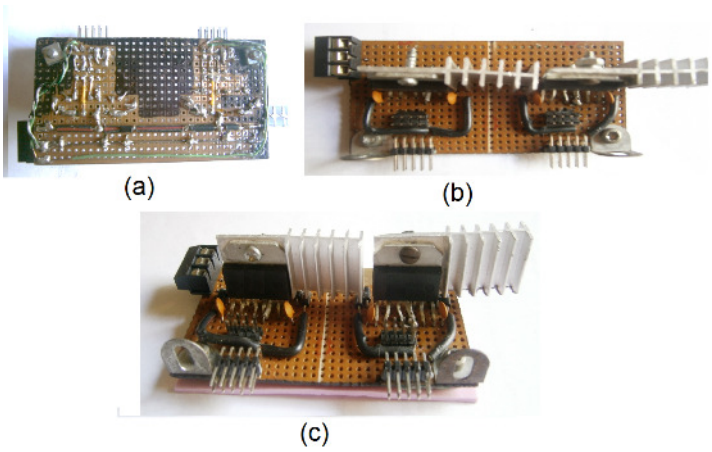


Figura 4.13: Fotografías de la implementación del módulo controlador de dos motores. Vistas: (a) inferior (b) superior (c) perfil.

4.3. Descripción del firmware

El firmware constituye la programación del único componente programable, el PIC18F2550, que es el núcleo de la Unidad de Control Cinemático (UCC).

La programación inherente a él fue simplificada para su explicación en este capítulo. El código fuente definitivo puede verse en el apéndice.

4.3.1. Diagrama conceptual de la programación

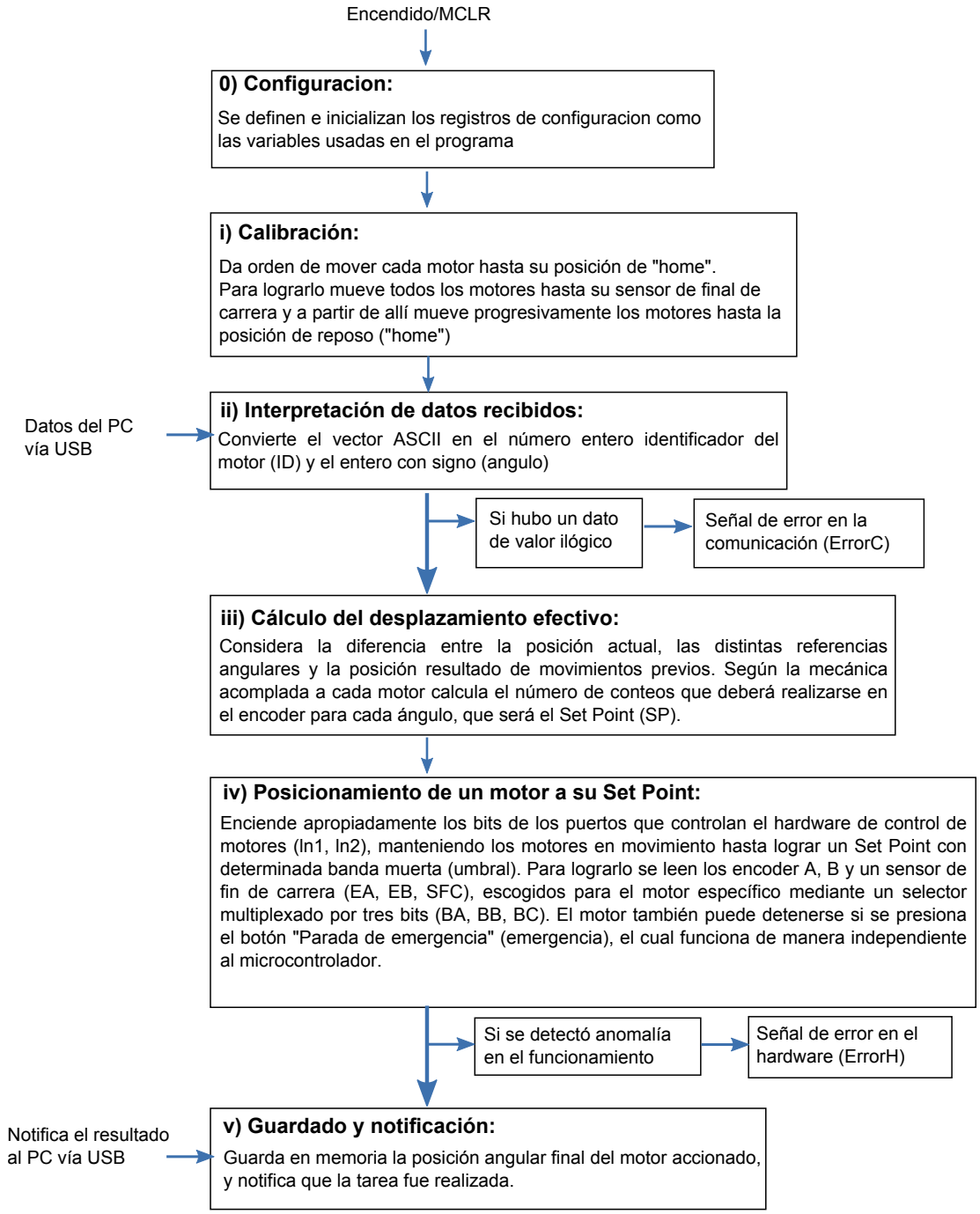


Figura 4.15: Diagrama conceptual de la programación.

4.3.2. Diagrama de flujo simplificado

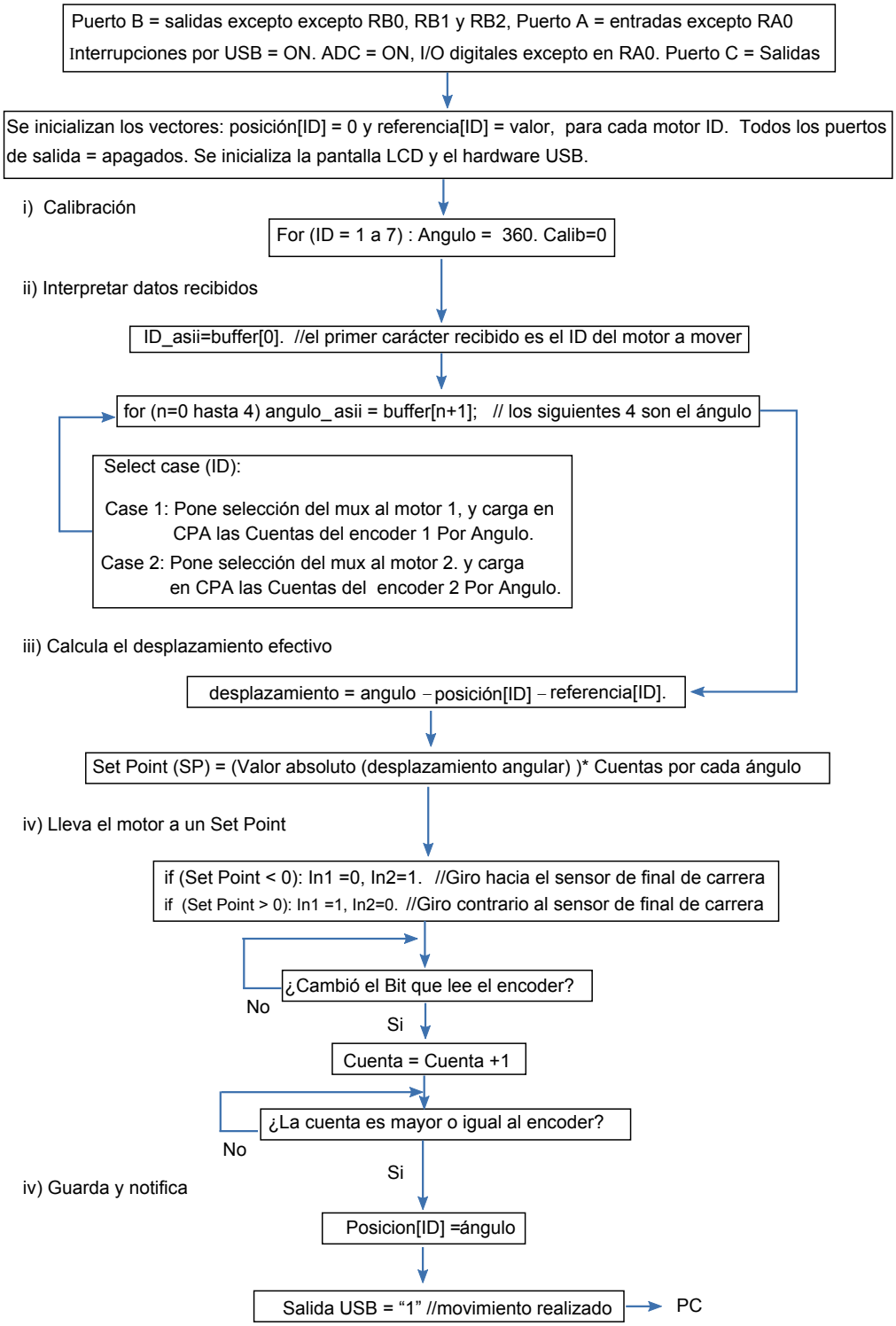


Figura 4.16: Diagrama de flujo simplificado.

4.3.3. Código fuente simplificado.

```
// 0)  configuraciones:
void interrupt( void ){
 bit estado;
    if (INTCON.INT0IF==1) {
   GIE_bit=0;
   estado = PORTB.B0;
   DELAY_US(20);      //mínimo 20Us, ruido de la
fuente conmutada PC
   if (estado == PORTB.B0){
  cuenta++;

Hid_Enable(&Buffer,&Bw);

while(1){
  n = Hid_Read();
          if (n !=0){
          ID_asii=buffer[0];
          ID = ID_asii-0x30;
// ii) interpretacion de datos recibidos
          for    (n=0;n<4;n++):    angulo_assii[n]    =
buffer[n+1];
          angulo = strToInt(angulo_assii,3);
//
int strToInt(char *string,char tamano)   // "String =
-125" menos ciento veinticinco grados.
{
 //Se definen e inicializan las variables internas del
programa
 int output = 0, i = 1, temp = 0, j;
  //Conversion a numero

for(j=1;j<tamano+1;j++)
 {
   temp = string[j] - 48;
  if (temp >= 0 && temp <= 9)
  {
   output += temp * pow(10, (tamano-i)); //output
= output + numero_leido * 10^(posicion)
    i++;
    }else if (temp ==0): output= output-1;
   }
 output = output + tamano-1;  //calibración
 //SIGNO
 if (string[0]==0x2B){        //Asii "+"
 }else if(string[0]==0x2D){  //Asii "-"
 output*=(-1);

       Bfreno=1, Bpwm =0;  //Se asegura que este
apagado todo (Freno =1 PWM=0)
       Posicion1 = Posicion[ID];
        Referencia1 = Referencia[ID];
         //CBA activo bajo
          if (ID==1){     //F //MUX 001
           SC=1; SB=1; SA=0; CPA=0x00;
          }else if (ID==2){ //E //MUX 010
           SC=1; SB=0; SA=1; CPA= 0x06;
          }else if (ID==3){   //D //MUX 011
           SC=1; SB=0; SA=0; CPA=0x08;
          }else if (ID==4){  //C //MUX 100
           SC=0; SB=1; SA=1; CPA=0x09;
          }else if (ID==5){   //B //MUX 110
           SC=0;  SB=0; SA=1; CPA=0x03;
          }else if (ID==6){   //A //MUX 001
           SC=1; SB=1; SA=0; CPA=0x03;
           // default: Message("Invalid state!");
          }

//  iii) Calcula el desplazamiento efectivo
          desplazamiento    =    angulo-posicion1-
referencia1;

          if (desplazamiento ==0){
          }else if (desplazamiento < 0){
          Bsentido = 1;
          }else if (desplazamiento > 0){
          Bsentido = 0;
          }
          temporal = desplazamiento * CPA;
          SP = abs(temporal);

iV) lleva el motor a su Set Point
          cuenta=1;
          RUN_PWM();
         fin=0;
         cuenta=0;
         while (fin==0){
         if(SP<cuenta)fin =1;
         if(porta.b5==0) fin=1;
         }
           Bfreno=1;
           Bpwm =0;

         inttostr(cuenta,tcuenta);
         lcd_out(2,11,tcuenta);

          //PWM_stop.

v) Guarda cambio y notifica
         Posicion[ID] = angulo;
         Bw[0]=0x01;
         porta=0xff;
          while(!HID_Write(&Bw,64));
          }
   }
}
```

Figura 4.17: Código fuente simplificado.

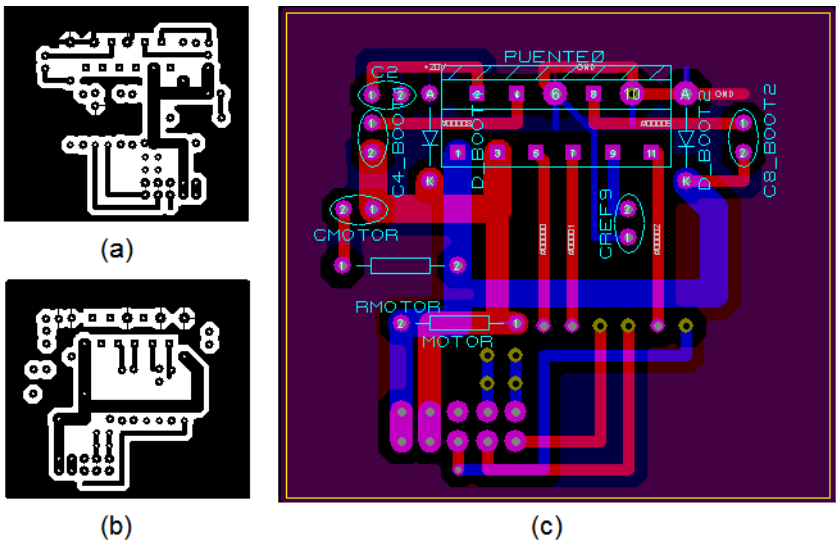


Figura 4.14: PCB del módulo de control de motores en Ares Proteus con L293B. Vistas: (a) Superior (b) Inferior. (c) Superpuestas superior rojo, inferior azul.

5 Diseño del software de control

5.1. Estudio Cinemático Directo

En el problema de la Cinemática Directa, dada una posición inicial del efector final del robot, se desea determinar su posición luego de aplicar movimientos de traslación y rotación que dependen de la configuración particular de la articulación. De acuerdo con el estudio matemático realizado en la sección 2.7.1, y luego de haber obtenido la matriz de transformación del Rhino XR-3 y sus ecuaciones respectivas, se aplican identidades trigonométricas correspondientes a la suma y diferencia de ángulos, y se pueden simplificar las expresiones, obteniendo así el siguiente modelo cinemático directo que se presenta a continuación:

$$T = \begin{bmatrix} nx & sx & ax & px \\ ny & sy & ay & py \\ nz & sz & az & pz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$nx = C1C(\theta_2 + \theta_3 + \theta_4)C5 - S1S5 \tag{5.1}$$
$$ny = S1C(\theta_2 + \theta_3 + \theta_4)C5 + C1S5 \tag{5.2}$$
$$nz = S(\theta_2 + \theta_3 + \theta_4)C5 \tag{5.3}$$

$$sx = -C1C(\theta_2 + \theta_3 + \theta_4)S5 - S1C5 \tag{5.4}$$
$$sy = -S1C(\theta_2 + \theta_3 + \theta_4)S5 + C1C5 \tag{5.5}$$
$$sz = -S(\theta_2 + \theta_3 + \theta_4)S5 \tag{5.6}$$

$$ax = -C1S(\theta_2 + \theta_3 + \theta_4) \tag{5.7}$$
$$ay = -S1S(\theta_2 + \theta_3 + \theta_4) \tag{5.8}$$
$$az = C(\theta_2 + \theta_3 + \theta_4) \tag{5.9}$$

$$px = C1[LbC2 + LcC(\theta_2 + \theta_3) - LdS(\theta_2 + \theta_3 + \theta_4)] \quad (5.10)$$

$$py = S1[LbC2 + LcC(\theta_2 + \theta_3) - LdS(\theta_2 + \theta_3 + \theta_4)] \quad (5.11)$$

$$pz = LdC(\theta_2 + \theta_3 + \theta_4) + LcS(\theta_2 + \theta_3) + LbS2 + La \quad (5.12)$$

5.2. Estudio Cinemático Inverso

A partir del modelo matemático obtenido mediante el Algoritmo de Denavit-Hartenberg, para dar solución al problema cinemático directo presentado en la Sección 2.7.2, se busca plantear un sistema con igual numero de ecuaciones que de incógnitas, y así poder encontrar los valores de θ. Observando las ecuaciones (5.10) y (5.11) es sencillo determinar a θ_1 al dividirlas:

$$\theta_1 = Tg^{-1}(py/px)$$

Para hallar los valores del resto de los ángulos se apelará a estudios gráficos, trigonométricos y analíticos.

Observando el plano de trabajo y las condiciones en las que el robot funcionará, se estableció el parámetro R, que se define como la distancia del origen del sistema de referencia al punto (px,py) en el plano XY.

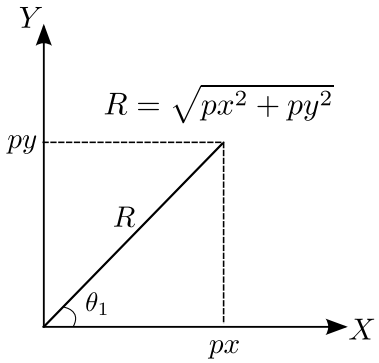


Figura 5.1: Representación del punto px,py en función de R.

El objetivo es hallar una relación para obtener $\theta_2, \theta_3, \theta_4$ dados R y z. Partiendo de las ecuaciones (5.10), (5.11) y (5.12), y usando la definición de R, se obtiene

dicha relación independiente de θ_1:

$$R = \sqrt{px^2 + py^2} = LbC2 + LcC(\theta_2 + \theta_3) - LdS(\theta_2 + \theta_3 + \theta_4) \tag{5.13}$$

$$pz = LdC(\theta_2 + \theta_3 + \theta_4) + LcS(\theta_2 + \theta_3) + LbS2 + La \tag{5.14}$$

Teniendo el sistema de ecuaciones anterior, se observa que hay 2 ecuaciones con 3 incógnitas, θ_2, θ_3 y θ_4. Sin embargo analizando el resto de las ecuaciones resultantes del modelo cinemático directo, se tiene la Ecuación (5.9) que define a la variable az, la cual indica la forma en que la herramienta «ataca» o enfrenta el plano de trabajo. Para la aplicación que se le dará al brazo antropomórfico XR-3, la herramienta siempre atacará al plano de trabajo en la dirección $-\hat{a}_z$, ya que se busca tomar objetos y llevarlos a puntos que se encuentren en el plano XY a una z determinada por el ambiente de trabajo y las dimensiones de la base del robot (ver Figura 5.2). De esta manera:

$$az = -1 = Cos(\theta_2 + \theta_3 + \theta_4) \tag{5.15}$$

La función coseno se hace igual a -1 para valores de $\pi + 2k\pi$; de esta forma se tiene para $k = 0$:

$$\theta_2 + \theta_3 + \theta_4 = \pi \tag{5.16}$$

sustituyendo el valor explicito de la ecuación (5.16) en las ecuaciones (5.10) y (5.12), se obtiene un sistema de 2 ecuaciones y 2 incógnitas:

$$\begin{aligned} R &= LbCos(\theta_2) + LcCos(\theta_2 + \theta_3) \\ pz &= -Ld + LcSen(\theta_2 + \theta_3) + LbSen(\theta_2) + La \end{aligned} \tag{5.17}$$

Se observa que se está en presencia de un sistema de ecuaciones no lineales, por lo que se utilizará la herramienta ***fsolve*** de MATLAB para hallar los valores de θ_2, θ_3. Conociendo a θ_2, θ_3, se calcula a θ_4 a partir de la ecuación (5.16).

El vector $\overrightarrow{s}$ indica el sentido de trabajo de la herramienta. sx ,sy y sz representan las componentes del vector $\overrightarrow{s}$ respecto al sistema de referencia fijo. El uso que se le dará a la herramienta se limitará, a tomar objetos del plano XY a una determinada Z. Por lo tanto, y de acuerdo a como se observa en la Figura (5.2) el sentido de apertura y cierre ($\overrightarrow{s}$) solo poseerá componente sx y sy , siendo así la componente $sz = 0$.

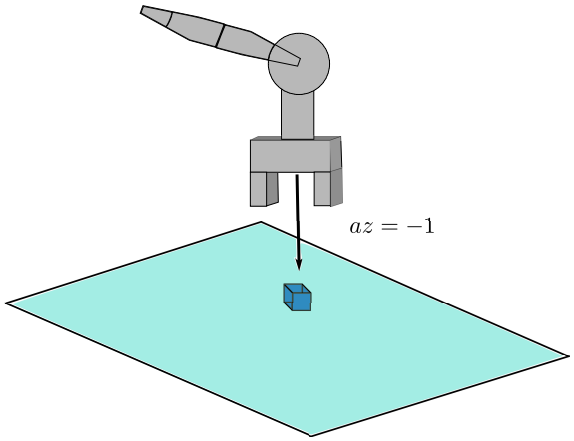


Figura 5.2: Herramienta de trabajo atacando el plano XY.

De acuerdo con lo expuesto anteriormente, se utilizarán las ecuaciones (5.4) y (5.5) para el calculo de θ_5. Mediante la visión por computador se hallaran sx y sy.

Utilizando las ecuaciones (5.4), (5.5) y (5.15) :

$$\begin{aligned} sx &= C1S5 - S1C5 = Sen(\theta_5 - \theta_1) \\ sy &= S1S5 + C1C5 = Cos(\theta_5 - \theta_1) \end{aligned} \tag{5.18}$$

Dividiendo las ecuaciones anteriores se tiene finalmente:

$$\theta_5 = Tg^{-1}(sx/sy) + \theta_1 \tag{5.19}$$

La dificultad de obtener modelos cinemáticos inversos se debe a la necesidad de despejar variables que se encuentren en el argumento de funciones trascendentes. En casos como el del robot Rhino XR-3 la dificultad se hace mayor porque todas las Matrices de Paso Homogéneas tienen como parámetros variables los ángulos de las articulaciones rotoides, de manera que el manejo de funciones *Seno* y *Coseno* es recurrente. Sin embargo estas expresiones no siempre arrojarán valores únicos y algunas veces resultaran inexistentes para algunas posiciones y orientaciones que no puedan ser definidas en la practica.

5.3. Ecuaciones del modelo cinemático

Las ecuaciones que definen al modelo cinemático resultado del estudio realizado son las siguientes:

Cinemática Directa:

$$R = LbCos(\theta_2) + LcCos(\theta_2 + \theta_3) \tag{5.20}$$
$$pz = -Ld + LcSen(\theta_2 + \theta_3) + LbSen(\theta_2) + La \tag{5.21}$$
$$px = RCos(\theta_1) \tag{5.22}$$
$$py = RSen(\theta_1) \tag{5.23}$$

Cinemática Inversa:

$$\theta_1 = Tg^{-1}(py/px) \tag{5.24}$$
$$\theta_4 = \pi - (\theta_2 + \theta_3) \tag{5.25}$$
$$\theta_5 = Tg^{-1}(sx/sy) + \theta_1 \tag{5.26}$$

$$\begin{cases} R &= LbCos(\theta_2) + LcCos(\theta_2 + \theta_3) \\ pz &= -Ld + LcSen(\theta_2 + \theta_3) + LbSen(\theta_2) + La \end{cases} \tag{5.27}$$

5.3.1. Diseño de algoritmo en MATLAB para evaluar el modelo cinemático inverso

Una vez que se obtienen las expresiones que permiten conocer los valores de los ángulos θ_1, θ_2, θ_3, θ_4 y θ_5, se diseñan dos algoritmos que modelan en gráficos 2D y 3D el funcionamiento del robot de una manera simple, para visualizar la llegada apropiada del brazo a un determinado punto.

El primer algoritmo trabaja con los valores de θ_1, θ_2, θ_3, θ_4 y un punto $P(x, 0, z)$. En primer lugar este se encarga de visualizar una perspectiva del robot en el plano XZ, donde los puntos $p1, p2, p3, p4$ representaran las articulaciones, y conociendo la longitud que existe entre una articulación y la siguiente se puede construir una representación gráfica en 2D, conformada solo por puntos, y lineas, como se observa en la Figura 5.3

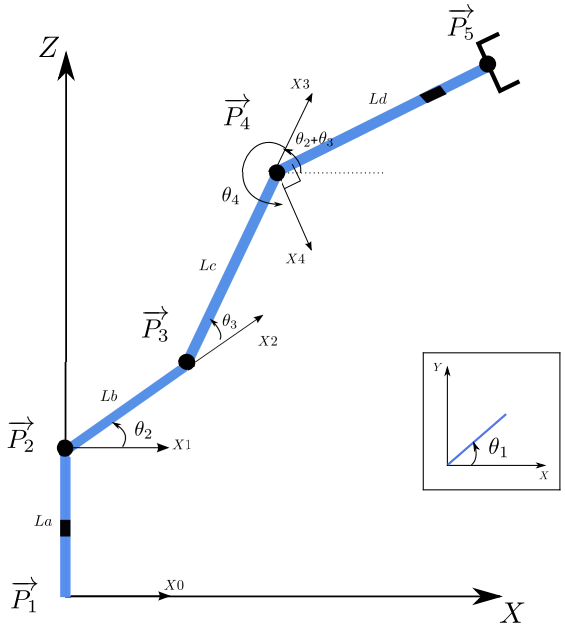


Figura 5.3: Representación en el plano XZ del Rhino XR-3 con sus respectivos ángulos.

En función de la Figura 5.3 se pueden escribir los puntos:

$$\overrightarrow{P_2} = La\,\hat{a_x}$$

$$\overrightarrow{P_3} = \overrightarrow{P_2} + Lb\,[Cos\theta_2\hat{a_x} + Sen\theta_2\hat{a_z}]$$

$$\overrightarrow{P_4} = \overrightarrow{P_2} + Lc\,[Cos(\theta_2+\theta_3)\hat{a_x} + Sen(\theta_2+\theta_3)\hat{a_z}]$$

$$\overrightarrow{P_5} = \overrightarrow{P_4} + Ld\left[Cos(\theta_2+\theta_3+\theta_4+\tfrac{\pi}{2})\hat{a_x} + Sen(\theta_2+\theta_3+\theta_4+\tfrac{\pi}{2})\hat{a_z}\right]$$

Para trasladar los puntos anteriores $\overrightarrow{P_i}$ en el plano XZ, a puntos $\overrightarrow{Q_i}$ en el espacio XYZ se utiliza θ_1 de la siguiente forma:

$$\overrightarrow{Q_i} = \left[\overrightarrow{P_i} \circ \hat{a_x}Cos(\theta_1)\right]\hat{a_x} + \left[\overrightarrow{P_i} \circ \hat{a_x}Sin(\theta_1)\right]\hat{a_y} + \left[\overrightarrow{P_i} \circ \hat{a_z}\right]\hat{a_z}$$

Luego de esto se elabora un código que se encarga de unir los puntos, y de crear pasos cortos de variaciones de cada angulo para que se pueda apreciar el posicionamiento progresivo del robot. Al proporcionarle a este algoritmo valores de ángulos, se podrá observar en modo de animación en 3D como el robot se posiciona en un punto especifico.

5.3.1.1. Evaluación del modelo cinemático inverso

Una vez implementado el algoritmo diseñado en la sección 5.3.1, se obtiene como resultado una representación gráfica de como se movería el Rhino XR-3 para llegar a cualquier punto dentro de su área de trabajo. En la Figura 5.4 se el cuadro final de la animación que muestra como el brazo robótico llega a un $P(x, y, z)$, comprobando que el planteamiento y resolución del modelo cinemático inverso es el correcto (ver Apéndice 7.4). .

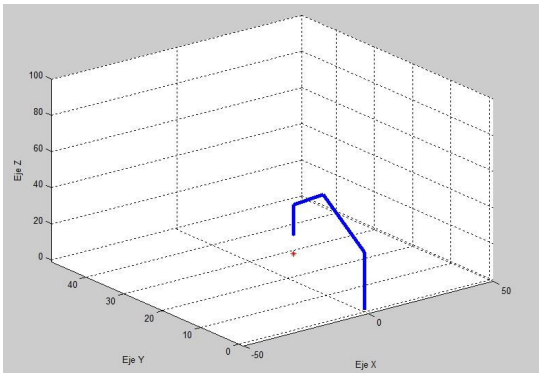


Figura 5.4: Dibujo en 3D generado por MATLAB para comprobar modelo cinemático inverso.

5.3.2. Diseño del algoritmo en MATLAB que desarrolla la cinemática directa

Para obtener los valores de un punto $P(x, y, z)$ dados unos θ_1, θ_2, θ_3, θ_4 y θ_5, se crea la función:

```
function [ x,y,z ] = Cinematicadirecta( theta1,theta2,theta3)
```

Se observa que el modelo matemático expuesto en la Sección 5.3 muestra al punto $P(x, y, z)$ independiente de θ_4 y θ_5, por lo tanto el algoritmo hace uso de las ecuaciones 5.20, 5.21, 5.22 y 5.23 y se obtienen px, py y pz. Este algoritmo tiene incidencia exclusiva a nivel de interfaz ya que al hardware del Rhino XR-3 son enviados sólo los ángulos.

5.3.3. Diseño del algoritmo en MATLAB que desarrolla la cinemática inversa

Haciendo uso del conjunto de ecuaciones 5.24, 5.25, 5.26 y 5.27 se hace el diseño de la siguiente función:

```
function [vt1 vt2 vt3 vt4] = Vectorthetas(vx,vy,vz)
```

La cual recibe los valores de px, py, y pz en forma de vectores, los que corresponden a cada punto de la trayectoria que se requiera. A su vez ésta hace uso de la función

```
function [theta1,theta2,theta3,theta4] = calculothetas(x,y,zz)
```

Quien a través de la función *fsolve* descrita en la Sección 2.8.2.1 resuelve el sistema de ecuaciones formado por 5.27, y devuelve como resultado los valores de θ_2 y θ_3, mientras que los valores θ_1 y θ_4 se obtienen con las Ecuaciones 5.24 y 5.25. Finalmente el valor θ_5 se obtiene a partir del procesamiento de imágenes explicado con detalle en la Sección 5.6.3 , (ver Apéndice 7.3).

5.4. Visión por Computador

Para dotar al Rhino XR-3 de visión artificial, se utiliza una cámara web, la cual proporciona la imagen a analizar, y luego de realizar una serie procesos que permitan la interpretación apropiada de la escena, se logra que el robot pueda moverse automáticamente y tomar un objeto de determinado color y trasladarlo a un punto en el plano de trabajo. En la Figura 5.5 se muestra un esquema que explica de manera sencilla como trabaja la Visión Artificial incorporada al funcionamiento general del Rhino. En el diagrama de la Figura 5.6 se muestran las diferentes acciones a realizar para lograr un exitoso funcionamiento de la Visión Artificial.

5.4.1. Representación de la imagen en el espacio.

Para trabajar con la Visión Artificial, y poder obtener imágenes que proporcionen información acerca de escenas específicas, el primer paso debe ser conocer el dispositivo que permita hacer esto posible. Como dispositivo de adquisición de imágenes, se utiliza una cámara web, la cual posee un conjunto de resoluciones de operación. En el presente trabajo, se decidió utilizar una cámara web

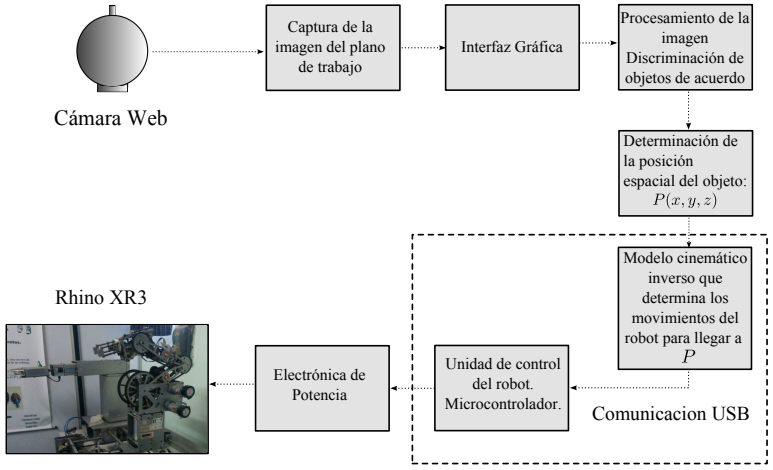


Figura 5.5: Esquema de funcionamiento general.

que proporciona imágenes con una relación de aspecto de 3:4, correspondiente a una resolución de 640×480 píxeles.

En cuanto a la posición de la cámara, el enfoque mas usado y el utilizado en este caso, es el modelo de cámara de un punto, donde el centro óptico es considerado el centro de la imagen, de forma que todos los puntos en el aparato de la imagen son dibujados por la luz que viaja desde el objeto del mundo real al punto central, hasta llegar a la cámara, como se observa en la Figura 5.7. Los ángulos α y β relacionan el sistema coordenado de la cámara a un sistema coordenado del mundo real, y especifican su posición y orientación en el espacio.

Tomando en cuenta lo anterior, y la aplicación que se le dará al sistema de Visión Artificial que se desea implementar, se podría hacer una representación gráfica de la escena a través de la Figura 5.7.

De acuerdo con la Figura 5.7 se deducen las siguiente ecuaciones:

$$\begin{aligned} X &= 2Z\,Tan(\alpha)Cos(\beta) \\ Y &= 2Z\,Tan(\alpha)Sen(\beta) \end{aligned} \tag{5.28}$$

Estas relaciones se deben a que siempre se observará un rectángulo de proporción 3:4 en la imagen capturada por la cámara, sin importar la altura de observación. β es fijado por la geometría a utilizar, mientras que α depende de la mecánica óptica del dispositivo y representa la apertura máxima a la cual la cámara captura toda la imagen sin distorsión.

$$\beta = Tan^{-1}(Y/X) \qquad \alpha = Tan^{-1}(d/Z) \tag{5.29}$$

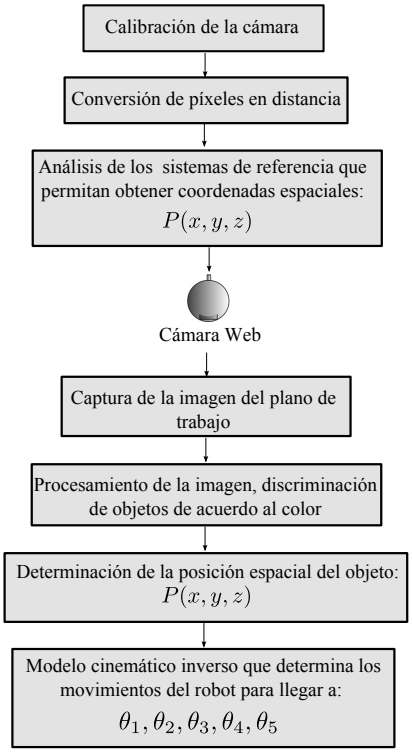


Figura 5.6: Esquema de Funcionamiento de la Visión Artificial.

Se realizan varios ensayos para recrear la escena de trabajo, y así determinar los valores de X, Y, Z con los que se trabajarán. Se miden las dimensiones reales del rectángulo visto en la imagen (valores X e Y) para una distancia de observación conocida (Z), y con la condición que el eje focal de la cámara se encuentre perpendicular al plano observado, y que la imagen se aprecie en su totalidad, es posible obtener finalmente β y α [30].

5.4.2. Conversión de píxeles en distancia.

En cualquier problema de Visión Artificial se trabaja con dos sistemas de coordenadas completamente diferentes: el sistema de coordenadas de la imagen captada, y el sistema de coordenadas de la escena. El sistema de coordenadas de la imagen es el que maneja la mayoría de los algoritmos de proceso y reconocimiento, es de dos dimensiones y su unidad es el píxel. El sistema de coordenadas de la escena suele ser tridimensional y sus unidades serán las clásicas unidades de longitud, como: metros, milímetros o pulgadas [19].

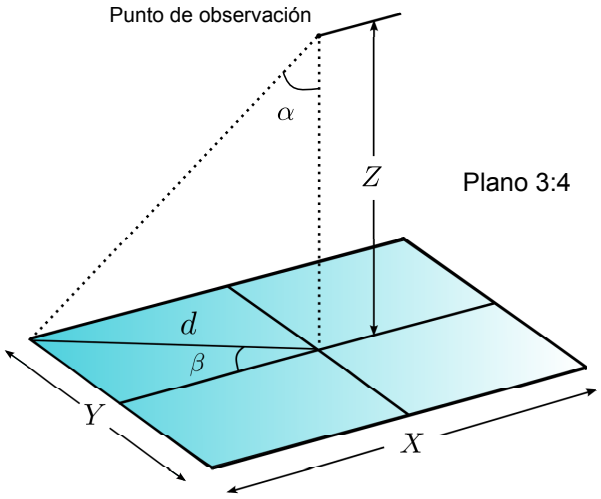


Figura 5.7: Geometría para la calibración de la cámara
Fuente: T. Rojas (2008)[30]

Para determinar las coordenadas espaciales de los objetos detectados en una imagen, se debe conocer el valor de la ubicación del objeto en píxeles, de esta manera se transforma la posición del píxel central de un objeto (coordenadas en cantidad de píxeles respecto al origen de la imagen) en una distancia de valor relativo a las dimensiones conocidas del plano observado. [30]

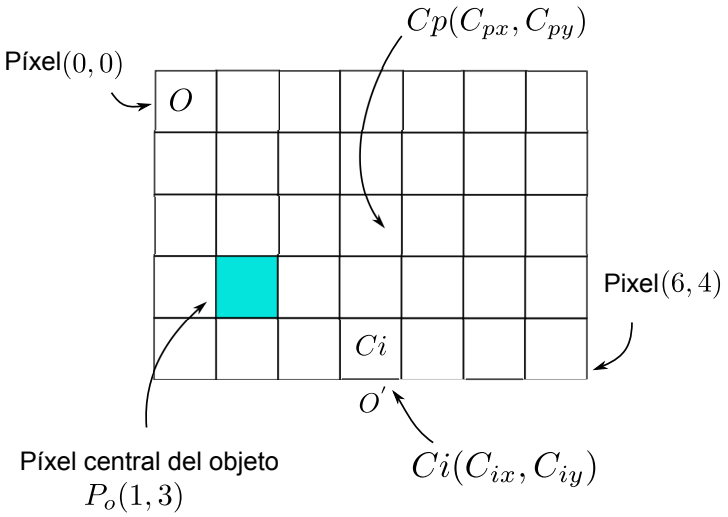


Figura 5.8: Imagen con un objeto detectado
Fuente: T. Rojas (2008)[30]

Dado, por ejemplo, el píxel $P_o(1,3)$ que representa el centro de un objeto detectado en la imagen (ver Figura 5.8), se observa que las coordenadas se encuentran referidas al origen O ubicado en la esquina superior izquierda de la imagen. De esta forma se requieren las coordenadas (P_{cx}, P_{cy}) del píxel P_o respecto al centro de la imagen $C_i = (C_{ix}, C_{iy}) = (3,4)$, el cual se definió de

esta forma para evitar variaciones en el signo de P_{cy}. Sin embargo hay que aclarar que a niveles de la posición de la cámara, ésta siempre coincidirá con el punto central del plano de observación de manera física $C_p = (C_{px}, C_{py})$, a pesar que este no sea considerado a niveles de cálculo como el centro del plano de observación. Con base en esto, se obtienen las siguientes expresiones:

$$P_{cx} = P_{ox} - C_{ix} = 1 - 3 = -2$$
$$P_{cy} = C_{iy} - P_{oy} = 4 - 3 = 1$$

A partir de la imagen de la Figura 5.8 como referencia, se plantea el siguiente conjunto de coordenadas para el centro del objeto:

-Respecto al origen de la imagen (O)

$P_o = (P_{ox}, P_{oy})$ en píxeles.

-Respecto al centro de la imagen (C_i)

$P_c = (P_{cx}, P_{cy})$ en píxeles.

-Respecto al centro del plano de observación $(O^{'})$

$P^{'} = (P_x^{'}, P_y^{'})$ en unidades de longitud.

Definiendo los siguientes valores:

1. Ancho de la Imagen (W_1) píxeles.
2. Largo de la imagen (H_1) píxeles.
3. Ancho del plano de observación (X) en unidades de longitud.
4. Largo del plano de observación (Y) en unidades de longitud.
5. Centro de la imagen $C = (INT(W_1/2), H)$ en píxeles.
6. Factor de imagen $F_1 = (Y/H_1) = (X/W_1)$ en unidades de longitud por píxel.

donde INT es la función «parte entera», se tiene que:

$$P_{cx} = P_{ox} - INT(W_1/2) \qquad P_{cy} = H_1 - P_{oy} \tag{5.30}$$

Finalmente en unidades de longitud se tiene:

$$P_x^{'} = P_{cx} \cdot F_1 \tag{5.31}$$
$$P_y^{'} = P_{cy} \cdot F_1$$

5.4.3. Obtención de las coordenadas espaciales.

Una vez obtenida la posición en dos dimensiones del objeto respecto al centro del plano de observación ($\overrightarrow{P_1}$), se pueden obtener las coordenadas de su ubicación espacial respecto al origen de cualquier sistema de referencia X, Y, Z. En este caso se necesita obtener la ubicación espacial respecto al sistema de referencia propio del Rhino XR-3 ($\vec{P'}$). De acuerdo con esto, se puede hacer una representación gráfica de la relación entre los dos sistemas como se muestra en la Figura 5.9. En esta, la altura del centro de un cuerpo de forma cúbica de lado (L) conocido respecto a su base es fácilmente calculable según la ecuación (ver Figura 5.9):

$$Z_{cubo} = Z_c - H_c + (L/2) \tag{5.32}$$

Finalmente y según la Figura 5.9, las coordenadas (X, Y) del cubo se determinan a partir de la suma vectorial:

$$\overrightarrow{P_{cubo}} = \overrightarrow{P_{centro}} + \overrightarrow{P_1} \tag{5.33}$$

donde $\overrightarrow{P_1}(P'_x, P'_y)$ son las coordenadas resultantes de la conversión de píxeles a distancias detallado en el punto anterior a través de la Ecuación (5.31).

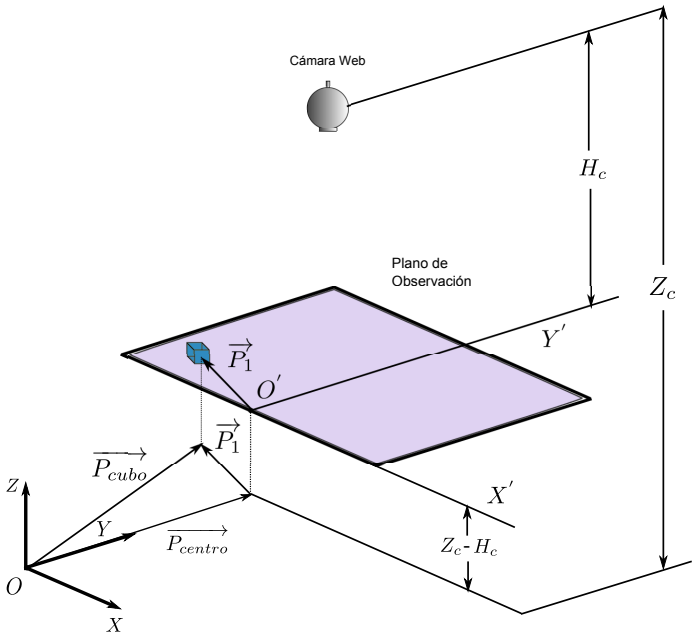


Figura 5.9: Perspectiva de la ubicación tridimensional del objeto.

5.5. Implementación de la visión por computador

El sistema de visión por computador a utilizar se muestra en la Figura 5.10, y esta compuesto por una cámara modelo ilook 300 de marca Genius, la cual se encuentra en una barra ajustable que permite definir la altura y posición de manera sencilla. La base que define el área de trabajo, fue elegida de color blanco para facilitar el procesamiento de la imagen que hace la captura de la escena, y que es procesada para identificar la ubicación espacial de el objeto de determinado color (rojo, verde o azul). Una vez determinada la ubicación espacial del objeto se procede a calcular cuales son los movimientos necesarios que debe realizar el Rhino para poder llegar al objeto y realizar las tareas que le sean indicadas.

El área elegida para trabajar con la visión artificial fue el resultado de la intersección del área del alcance del robot con la imagen de mayor dimensión que puede ser capturada por la cámara a una altura determinada, como se observa en la Figura 5.11

Figura 5.10: Sistema para la visión por computador.

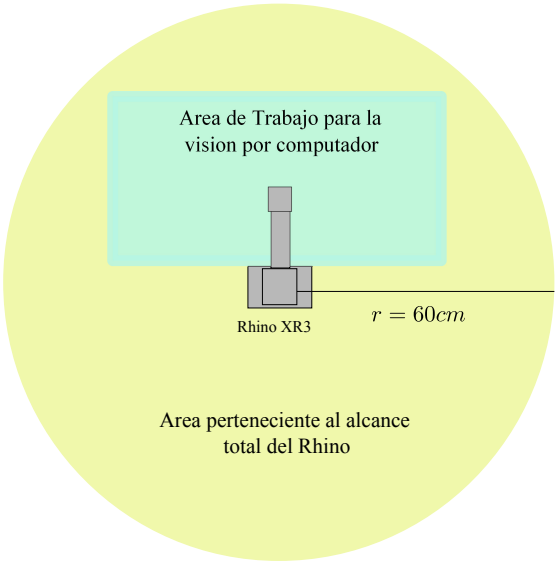


Figura 5.11: Área de Visión sobre el área de trabajo del Rhino XR-3.

5.5.1. Ubicación de los objetos en el espacio

5.5.1.1. Representación de la imagen en el espacio.

La cámara elegida para realizar la visión artificial fue de modelo ilook 300 de marca Genius, la cual se adecuó para trabajar con una resolución de 640x480 píxeles. De forma experimental se obtuvieron las dimensiones reales del rectángulo visto en la imagen de ancho $x = 54{,}1\,cm$, y largo $y = 39{,}4\,cm$, para una distancia de observación conocida de $Z = 81\,cm$, con la condición que el eje focal de la cámara se encuentre perpendicular al plano observado.

Recordando que β es fijado por la geometría a utilizar, mientras que α depende de la mecánica óptica del dispositivo, se sigue el procedimiento explicado con detalle en la Sección 5.4.1 para obtener finalmente los valores de los parámetros:

$$\beta = Tan^{-1}(Y/X) \qquad \alpha = Tan^{-1}(d/Z) \tag{5.34}$$

$$d = \sqrt{\left(\frac{x}{2}\right)^2 + \left(\frac{y}{2}\right)^2} = 33{,}4633\,cm$$

$$\beta = 36{,}0651^{o} \quad \alpha = 22{,}4468^{o}\,cm$$

5.5.1.2. Conversión de píxeles en distancia.

Para lograr el correcto posicionamiento de la herramienta del Rhino, y que éste sea capaz de tomar un objeto, deben realizarse los ajustes que sean necesarios para que a partir de los píxeles identificados como el centro del objeto en la imagen, se puedan obtener las coordenadas de ese punto en unidades de longitud, medidas respecto al sistema de referencia propio del brazo robot. Según las ecuaciones presentes en la sección 5.4.2 se tiene:

$$\begin{aligned} P_x^{'} &= P_{cx} \cdot F_1 \\ P_y^{'} &= P_{cy} \cdot F_1 \end{aligned} \tag{5.35}$$

Donde P_C representa el centro del objeto en píxeles referido al sistema de coordenadas de la imagen y F_1 el factor que relaciona píxeles con longitud, dando como resultado lo siguiente:

$$F_1 = (Y/H_1) = (39{,}4/480) = 0{,}0820\frac{cm}{pixel}$$

De este manera se transforman las coordenadas de un punto en píxeles obtenidas por el ***Algoritmo para punto central*** explicado en la Sección 2.8.2, referidos al sistema de coordenadas de la imagen en unidades de longitud $P_x^{'}$ y $P_y^{'}$, como se muestra en la Figura 5.12

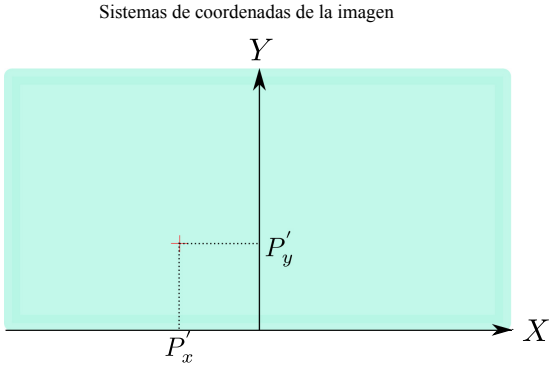


Figura 5.12: Sistema de coordenadas de la imagen.

5.5.1.3. Obtención de las coordenadas espaciales.

Para poder conocer los movimientos exactos que debe hacer el Rhino para ubicar su TCP en un punto especifico, a partir de la visión artificial, deben

conocerse a través del procesamiento de imágenes las coordenadas referidas al sistema de referencia del brazo. Para que esto sea posible se definió $\overrightarrow{P_{centro}}$ como el vector que apunta desde el sistema de referencia del robot hasta el sistema de referencia de la imagen, y Z_{objeto} como la altura del centro geométrico del objeto. De esta forma:

$$\overrightarrow{P_{centro}} = 20\hat{a_y} \quad Z_{objeto} = Z_c - 70cm + (L/2)$$

Una vez obtenido $\overrightarrow{P_{centro}}$ es posible calcular las verdaderas coordenadas espaciales de un objeto mediante la visión por computador:

$$\overrightarrow{P_{objeto}} = \overrightarrow{P_{centro}} + \overrightarrow{P_1} + Z_{objeto}\hat{a_z}$$

5.6. Procesamiento de Imágenes utilizando MATLAB

5.6.1. Detección de objetos.

Para la detección de objetos, se diseñó un algoritmo capaz de distinguir en una imagen un objeto de determinado color: rojo, verde, o azul. Se realiza la respectiva lectura de la imagen, y se almacena en el arreglo $img(i, j,)$, donde `color` adquiere valores según el color: si es rojo $= 1$, verde $= 2$, y azul $= 3$. Una vez determinado el color que se desea detectar, se recorre píxel a píxel, lo cual es equivalente a recorrer las 3 matrices de dos dimensiones, elemento a elemento. En este recorrido dependiendo del color, se evalúa si un píxel es considerado de rojo, verde o azul.

Para considerar que un píxel es de un determinado color se aplican dos condiciones. La primera consiste en establecer un valor de referencia llamado $fac1$ el cual es resultado de multiplicar el máximo valor de intensidad de toda la imagen por un valor de tolerancia tol establecido arbitrariamente $fac1 = tol \cdot \text{máx}(img(:, :, color))$, siendo $tol < 1$. Para considerar que un píxel es de un color dado (rojo, verde o azul) se debe cumplir que $img(i, j, color) > fac1$. Mientras mayor sea tol, mas se restringe la intensidad de un determinado color que puede tener un píxel.

La segunda condición verifica que efectivamente el píxel posee mayor intensidad de un color especifico (el color elegido), que de los otros dos. De esta manera,

para que un píxel sea considerado de un color dado, la intensidad de ese color debe ser mayor a la máxima intensidad de los otros dos colores, pesado por un valor denominado $fac2$, es decir $img(i, j, color) > fac2 \cdot \text{máx}(img(:, :, color2))$ y $img(i, j, color) > fac2 \cdot \text{máx}(img(:, :, color3))$. Así se asegura que ciertamente el píxel en estudio pertenece al objeto que se esta buscando. Mientras mayor sea $fac2$, mas restrictiva será la discriminación entre colores.

A medida que se discrimina si un píxel es del color seleccionado, se va creando progresivamente una nueva imagen binaria (C), donde los píxeles admitidos se colocan en 1, y los rechazados en 0. Posteriormente, la imagen binaria se retoca para mejorar el contorno que define al objeto eliminando ruido.

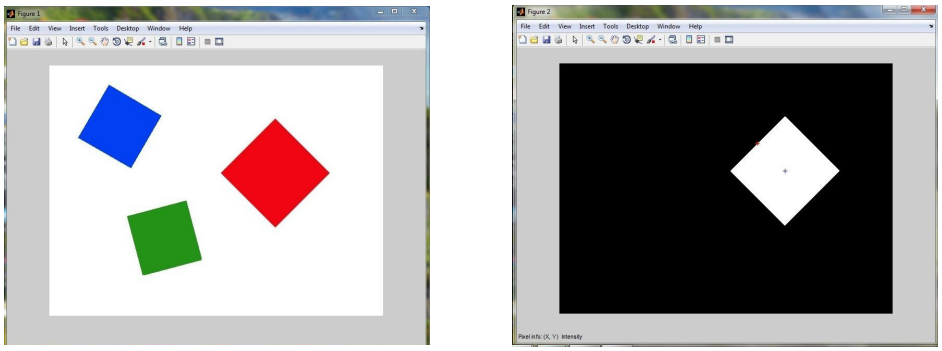
5.6.1.1. Implementación de la detección de objetos

El valor de la tolerancia tol que se define para esta aplicación se fija en 0,01 y $fac2$ se fija en 1,2 tras realizar una gran cantidad de pruebas, y comprobando que estos son los valores mas apropiados para hacer la detección de objetos. A medida que se discrimina si un píxel es del color seleccionado como se explicó detalladamente en la Sección 2.8.2, se crea progresivamente una nueva imagen binaria (C), donde los píxeles admitidos se colocan en 1, y los rechazados en 0. Posteriormente, la imagen binaria se retoca para mejorar el contorno que define al objeto eliminando ruido, utilizando ciertas operaciones y funciones del «*Image Processing Toolbox*» de MATLAB.

Una vez detectado el objeto se procede al calculo del centro geométrico del mismo, y del punto mas cercano al centro que pertenezca al objeto como se explica en la Sección 2.8.2, y están representados en las imágenes b) como una cruz azul, y una roja, respectivamente. La Figura 5.13 muestra como se realiza la detección de objetos cuando se trata con los colores de valor exacto: $Rojo = (255, 0, 0)$, $Verde = (0, 255, 0)$, $Azul = (0, 0, 255)$, y se observa que la detección del objeto es prácticamente exacta, ya que no existe distorsión por sombras o por otros factores que puedan dificultar la detección y hacerla menos precisa.

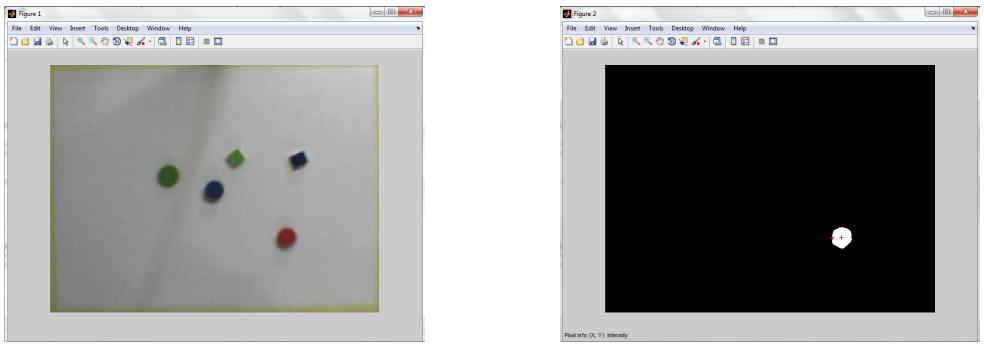
En las figuras, 5.14, 5.15 y 5.16 se observan imágenes reales tomadas en el área de trabajo del Rhino con los objetos diseñados para la aplicación, y se puede observar que la detección de objetos no es tan precisa como en la imagen de prueba. Sin embargo al variar un poco los valores de tol y de $fac2$ se

puede ajustar para evitar la distorsión en gran medida, mas no se elimina por completo (ver Apéndice 7.1).



a) b)

Figura 5.13: a) Imagen de prueba sin distorsión a causa de sombras b)Imagen procesada detectando a objeto de color elegido rojo con puntos de interés.

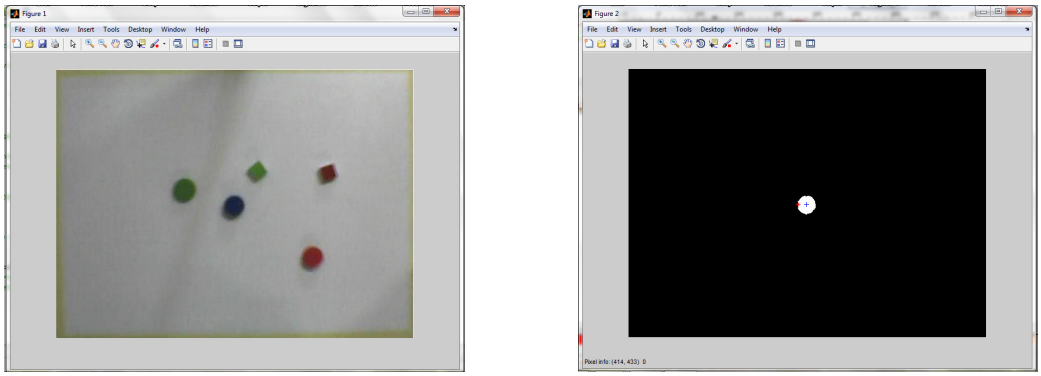


a) b)

Figura 5.14: Imagen real de la escena con distorsión a causa de sombras b)Imagen procesada detectando a objeto de color elegido rojo con puntos de interés.

5.6.2. Determinación de puntos de interés en la imagen.

La importancia de conocer la ubicación del centro del objeto se ha reseñado reiteradamente en los puntos anteriores de este capitulo. Para determinar las coordenadas del punto central, se diseñó un algoritmo basado en el promedio de posiciones de píxeles que pertenezcan al objeto. La imagen a aplicarsele dicho método es la imagen binaria C descrita en el punto anterior, previamente tratada para garantizar que no existan sombras o detalles que puedan dejar puntos aislados que no pertenezcan al objeto y ocasionar errores en la obtención de las



a) b)

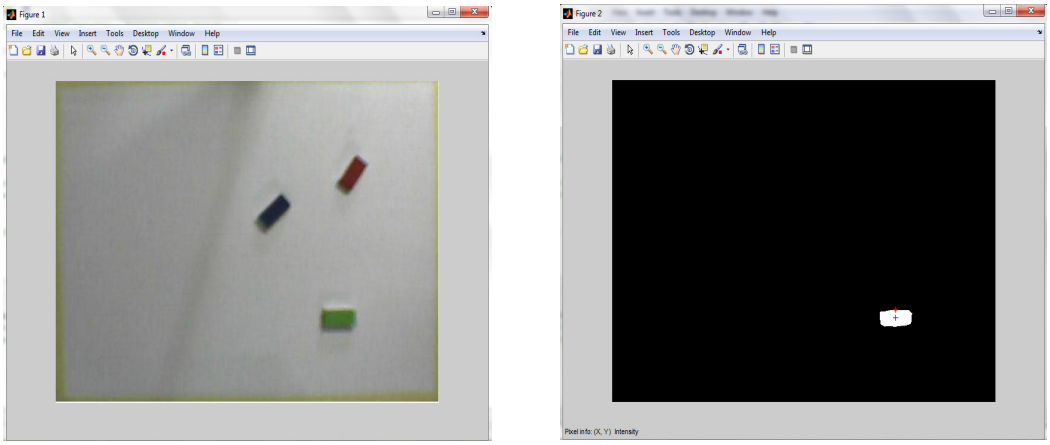
Figura 5.15: Imagen real de la escena con distorsión a causa de sombras b)Imagen procesada detectando a objeto de color elegido azul con puntos de interés.

componentes del centro geométrico en dos dimensiones del objeto. Las posiciones de los píxeles de la imagen, se acumulan por cada componente x y y cuando el píxel en cuestión posea una intensidad de 1 (blanco), y luego se dividen entre la cantidad N de píxeles blancos presentes. Realizando las conversiones a unidades de longitud, se obtiene el centro $\overrightarrow{P} = F_1 \left(\hat{a_x} \sum_{i=1}^{N} \frac{x_i}{N} + \hat{a_y} \sum_{i=1}^{N} \frac{y_i}{N} \right)$, siendo F_1 el factor de imagen para convertir píxeles en longitud.

Anteriormente se hizo referencia al calculo del ángulo θ_5 y se llegó a la conclusión de que para hallar dicho valor, se debe obtener $\overrightarrow{s}$. Para obtenerlo se crea un vector en dirección de $\overrightarrow{s}$, con origen en $\overrightarrow{P}$, y con extremo $\overrightarrow{J}$, siendo este el punto más cercano a $\overrightarrow{P}$ que se encuentra en el borde del objeto. El algoritmo diseñado para obtener las coordenadas de $\overrightarrow{J}$ consiste en calcular las distancias de cada píxel no perteneciente al objeto (utilizando la imagen binaria) a las coordenadas del centro del mismo. Luego se determina cual píxel posee la menor distancia, y las posición de ese píxel, representará, luego de las conversiones a unidades de longitud, las coordenadas de $\overrightarrow{J}$.

5.6.3. Determinación de las componentes sx y sy.

Utilizando el ***Algoritmo para punto central*** y el ***Algoritmo para punto cercano*** se hallaron los puntos $\overrightarrow{P}$ y $\overrightarrow{J}$ respectivamente. Debido a que siempre se busca tomar al objeto de la manera mas óptima, se requiere orientar la herramienta de forma que el robot lo tome por la zona de menor dimensión, y para garantizarlo se utilizarán estos dos puntos para crear al vector $\overrightarrow{PJ}$, que



a) b)

Figura 5.16: Imagen real de la escena con distorsión a causa de sombras b)Imagen procesada detectando a objeto de color elegido verde con puntos de interés.

va en la dirección apropiada para la apertura y cierre de la herramienta, y por lo tanto, $\overrightarrow{s} = \frac{\overrightarrow{PJ}}{|\overrightarrow{PJ}|} = s_x\hat{a_x} + s_y\hat{a_y}$ (ver Figura: 5.17).

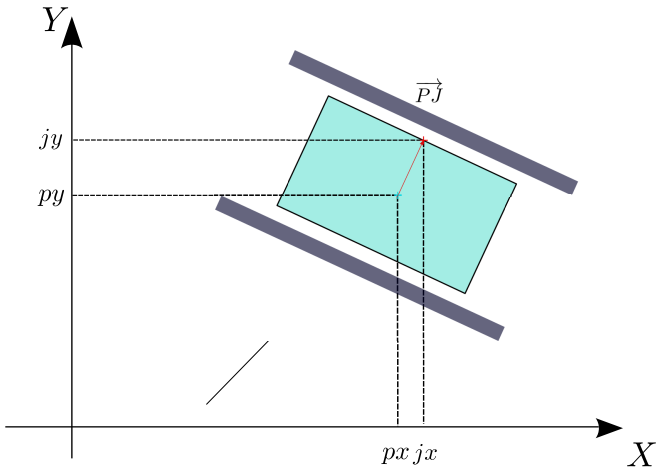


Figura 5.17: Representación del vector PJ ($\hat{PJ} = \overrightarrow{s}$).

5.7. Interfaz Gráfica

La interfaz gráfica que permite la manipulación del Rhino XR-3 desde el computador, consta de tres ventanas, una de inicio, una de Cinemática Directa y otra de Cinemática Inversa. En la Figura 5.18 se muestra el funcionamiento detallado de la Interfaz Gráfica.

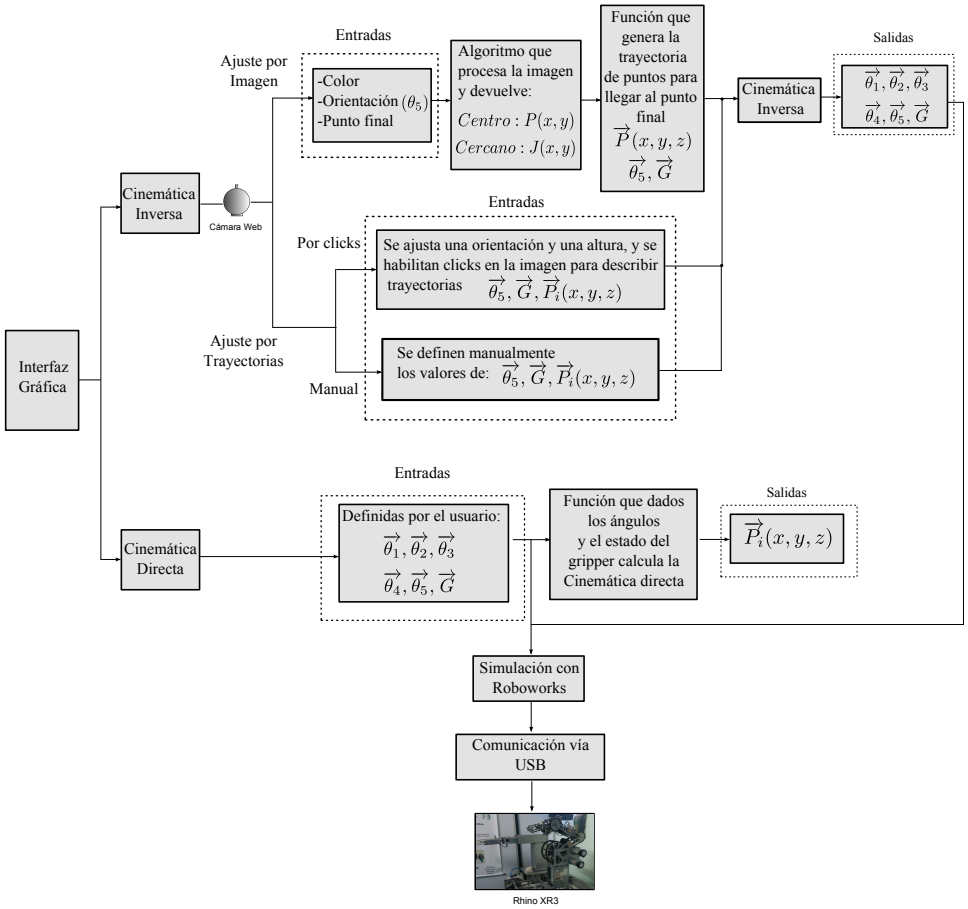


Figura 5.18: Esquema de Funcionamiento de la Interfaz Gráfica.

5.7.1. Ventana de Menú

La primera ventana, es la ventana de Menú, la cual muestra las dos opciones posibles para controlar al brazo antropomórfico como lo muestra la Figura 5.19. A través de esta ventana se puede acceder a través de botones a las dos ventanas restantes, las de Cinemática Inversa y Directa.

5.7.2. Ventana de Cinemática Inversa

La ventana de Cinemática Inversa consta de cinco secciones principales, donde se realiza la captura de la imagen del plano de trabajo, las áreas de Tareas Manuales y Automáticas, la zona de visualización, y la sección de acciones a tomar, sea comunicación o simulación.

Inicialmente al abrirse la ventana, la única opción habilitada para realizar es la de captura de la imagen a través de un botón. Una vez ejecutada esta acción, aparece dentro de la misma ventana, la imagen y adjunto a ella los ejes

Figura 5.19: Ventana de Menú.

coordenados correspondientes como lo muestra la Figura 5.20 (a).

Una vez que se tiene la foto, es posible escoger dos modos (excluyentes entre si) de ejecutar tareas, de forma Manual o Automática. La sección de Tarea Automática está estructurada de forma tal que el usuario al indicar el color del objeto que quiere ser manipulado por el Rhino XR-3, éste sea capaz de trasladarlo hasta un punto final que puede ser indicado en los campos de texto adjuntos como se muestra en la Figura 5.20 (b). La sección de Tareas Manuales se subdivide en dos áreas. El área de Ajuste por Clicks y el área de Ajuste Manual. El Ajuste por Clicks permite dar clicks sobre la imagen para generar la trayectoria que se desea que el Rhino XR-3 ejecute. Se tiene la opción de guardar valores de altura (Z), valores de θ_5 y estado del gripper (abierto/cerrado) iniciales que correspondan a los puntos generados, y si se desean modificar es posible hacerlo a través del Ajuste Manual. El Ajuste Manual, permite definir las coordenadas X,Y,Z,θ_5 y estado del gripper (abierto/cerrado) (ver Figura 5.20 (c)) para cada punto que se fije para ser seguido por el brazo antropomórfico. Estas dos áreas se encuentran vinculadas y se pueden añadir o modificar todos los parámetros de los puntos pertenecientes a trayectorias generadas por cualquiera de las dos modalidades.

De cualquiera de estas dos secciones se generan los vectores $\overrightarrow{X}$,$\overrightarrow{Y}$,$\overrightarrow{Z}$,$\overrightarrow{\theta_5}$,$\overrightarrow{G}$ listos para ser procesados y mediante el algoritmo descrito en la Sección 5.3.3 para obtener $\overrightarrow{\theta_1}$, $\overrightarrow{\theta_2}$, $\overrightarrow{\theta_3}$, $\overrightarrow{\theta_4}$, $\overrightarrow{\theta_5}$ y $\overrightarrow{G}$. Estos valores finales se muestran en la zona de resultados conformada por dos listbox que permiten la visualización de la lista de puntos y ángulos resultantes.

Una vez generados los ángulos que definan el movimiento del Rhino, se tiene la opción de llevar a cabo la simulación a través del software Roboworks, para

luego habilitar la Comunicación USB, ambas contenidas en el área de acciones (ver Figura 5.21).

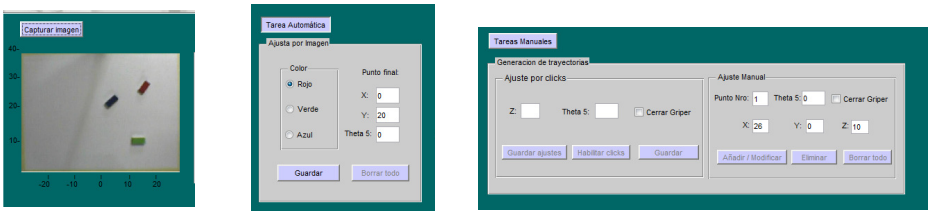


Figura 5.20: a)Área de captura de la imagen b) Área de Tarea Automática. c) Área de Tareas Manuales.

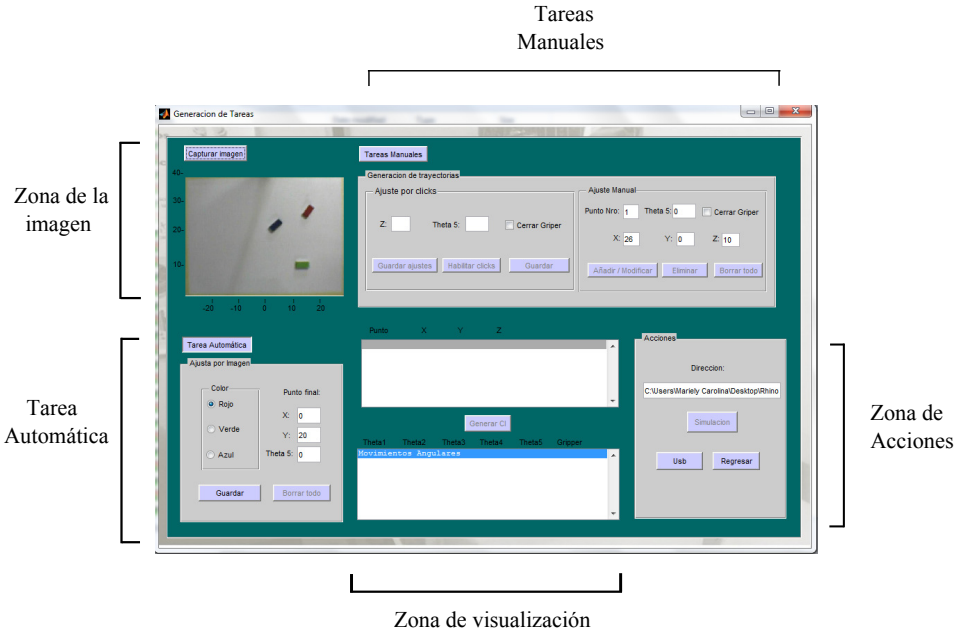


Figura 5.21: Ventana para la Cinemática Inversa.

5.7.3. Ventana de Cinemática Directa

La ventana de Cinemática Directa consta de tres secciones, zona de ingresos de ángulos, zona de visualización, y resultados y acciones. La zona de ingreso de ángulos permite la asignación de los ángulos correspondientes a cada articulación de forma manual a través de los campos de texto adjuntos. La zona de visualización permite observar la lista de movimientos para el robot, que están listos para ser utilizados por las distintas acciones disponibles. Por último se encuentra la zona de resultados y acciones, la cual muestra, una vez

ejecutado el algoritmo que permite la obtención de la cinemática directa (ver Sección 5.3.2), los valores de los vectores X, Y y Z. De esta misma forma es posible realizar la simulación para luego llevar a cabo la comunicación USB con el Rhino XR-3, (ver Figura 5.22).

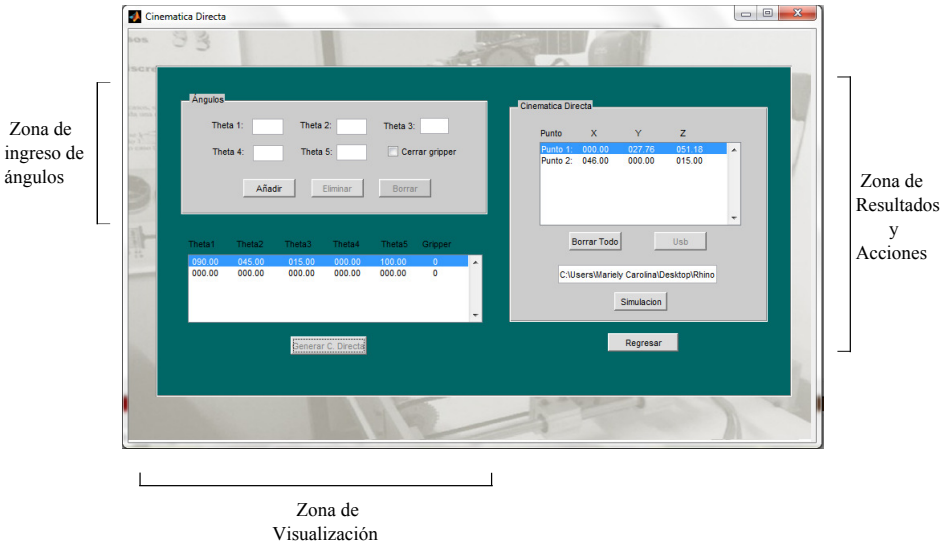


Figura 5.22: Ventana para la Cinemática Directa.

5.7.4. Orden del movimiento de las articulaciones

Una vez que se tienen definidos los vectores $\overrightarrow{\theta_1}$, $\overrightarrow{\theta_2}$, $\overrightarrow{\theta_3}$, $\overrightarrow{\theta_4}$, $\overrightarrow{\theta_5}$ y $\overrightarrow{G}$ se podría pensar que están listos para ser enviados a Roboworks, y luego a la comunicación USB, sin embargo, debe realizarse un último ajuste antes de ser transmitidos. Tomando en cuenta el funcionamiento del circuito electrónico que controla el brazo robot, el cual solo permite el movimiento de un servo-motor a la vez, surge la necesidad de determinar el orden apropiado en que cada articulación debe moverse para completar la trayectoria de un punto $P_i(x_i, y_i, z_i)$ a otro $P_f(x_f, y_f, z_f)$, sin generar inconvenientes que puedan comprometer el estado del Rhino o puedan ocasionar, por ejemplo, que algún brazo colisione con la superficie de trabajo. Se concluye, que puede definirse un orden específico diferente para los siguientes casos de movimiento de la herramienta: «subir», «bajar», «alejarse», «acercarse». Sin embargo, en general cualquier trayectoria entre dos puntos, presentará varios de estos casos simultáneamente, por lo cual se divide la trayectoria creando puntos intermedios, de manera que cada sub-trayectoria presente solo uno de dichos casos. Estos puntos intermedios se incluyen como parte de la trayectoria a realizar, de manera que queda plenamente determinado un orden de movimiento seguro, listo para ser enviado

tanto al Rhino como a Roboworks. En el Cuadro 5.1 se indica el orden de movimiento de las 5 articulaciones y el gripper, para cada uno de los 4 casos antes mencionados.

Por ejemplo, en la Figura 5.23 se observa que los cambios para realizar un movimiento desde P_i hasta P_f son dos: *alejarse* y *subir*, de acuerdo con esto se halla: un punto intermedio (Pa) que se encarga de posicionar al punto P_i en el plano XZ controlado por θ_1, y un segundo punto (Pb) que se encarga de *alejar* $(r_f > r_i)$, para luego *subir* $(z_f > z_i)$ y llegar a P_f (ver Apéndice 7.5).

Cuadro 5.1: Orden en que deben moverse las articulaciones del Rhino según el caso

Caso	Orden
Subir	$\theta_2, \theta_1, \theta_3, \theta_4, \theta_5, G$
Bajar	$\theta_5, \theta_4, \theta_3, \theta_1, \theta_2, G$
Alejarse	$\theta_1, \theta_3, \theta_2, \theta_4, \theta_5, G$
Acercarse	$\theta_5, \theta_4, \theta_2, \theta_1, \theta_3, G$

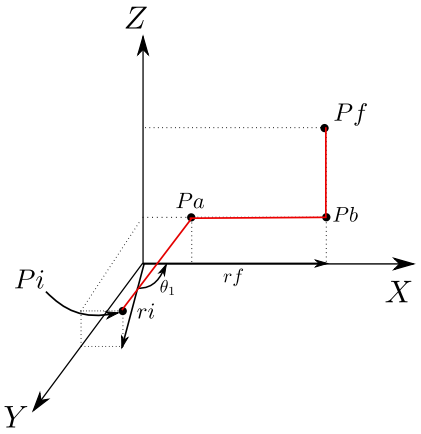


Figura 5.23: Caso específico para cálculo de puntos intermedios.

5.8. Diseño de algoritmo para la comunicación USB en MATLAB

Una vez que se tienen definidos los vectores definitivos de $\overrightarrow{\theta_1}$, $\overrightarrow{\theta_2}$, $\overrightarrow{\theta_3}$, $\overrightarrow{\theta_4}$, $\overrightarrow{\theta_5}$ y $\overrightarrow{G}$, y están listos para ser enviados, deben adecuarse a la estructura definida de la

comunicación USB. La comunicación USB Computador-PIC se da en forma de paquetes de datos de 64 bytes en cada envío. Se pueden enviar tantos paquetes como se requieran, sin embargo la limitación viene dada por la memoria RAM del PIC ya que cada paquete es almacenado en ella, y una vez recibidos todos, es que el PIC ejercerá su función de control.

Para convertir el valor de un ángulo en bytes que puedan ser enviados en paquetes, se debe hacer una transformación y descomposición del valor incluyendo el signo, la Figura 5.24 muestra como cada ángulo puede escrito en tres bytes de acuerdo a los parámetros ahí mostrados. Si se define que para determinar un movimiento completo del robot se deben conocer los valores de $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, G$, se puede decir que en bytes esto equivaldría a: $1 * angulo * 3(bytes/angulo) * 6 = 18bytes$. Si la capacidad por paquete a enviar es de 64 bytes, se podrían enviar 3 movimientos completos por cada paquete $3 * 18bytes = 54bytes$ como lo muestra la Figura 5.25. El primer byte del paquete es un 0, y esto se debe a que la librería HIDAPI así lo especifica para un correcta comunicación. En este caso particular para la aplicación del Rhino XR-3 la cantidad de paquetes a enviar estará delimitada hasta máximo 10 paquetes.

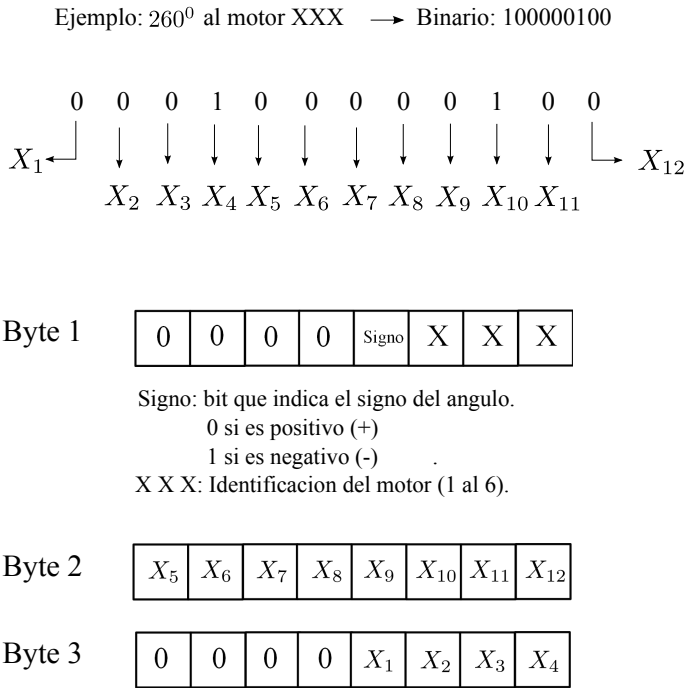
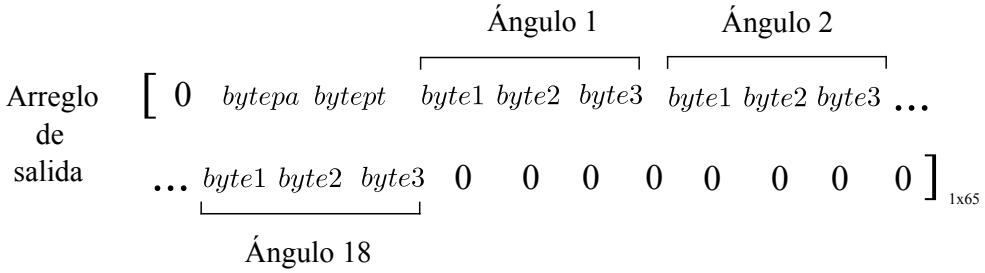


Figura 5.24: Transformación de un valor decimal con signo (ángulo) perteneciente a un motor en especifico.

Para generar el arreglo de salida anterior, primero se crea una función que



bytepa Byte que indica el numero de paquete actual.

bytept Byte que indica el numero de paquetes totales en la comunicación.

Figura 5.25: Estructura del arreglo salida de la comunicación USB.

dado el valor del angulo, esta devuelva los Byte 2 y Byte 3

```
function [byte2,byte3] = conversion(angulo)
```

Utilizando la función de matlab `valor=dec2bin(angulo)` se transforma el valor a un numero binario, el cual es nuevamente transformado en una cadena string (`num2str(valor)`), luego desglosado por caracter y concatenados nuevamente, pero con el orden requerido, utilizando la función `strcat`. Finalmente se utiliza la función `base2dec` para convertir el valor binario a uno entre 0 - 255, y así formar parte del arreglo de salida.

Posteriormente se crea una función que dado el valor del ángulo con signo, y el motor al que corresponde, genere el Byte 1, y dentro de ella el Byte 2 y 3. Esta función se encarga de generar la cadena de caracteres que tendrá la estructura correspondiente. De acuerdo al motor, se le asigna en los bits menos significativos el equivalente correspondiente, y ademas evalúa el signo del angulo para generar el Byte 1.

```
function [byte1,byte2,byte3 ] = paquete(angulo,motor)
```

Una vez obtenido el algoritmo que permite convertir un angulo en tres Bytes, se procede a hacer el recorrido de cada uno de los vectores $\overrightarrow{\theta_1}$, $\overrightarrow{\theta_2}$, $\overrightarrow{\theta_3}$, $\overrightarrow{\theta_4}$, $\overrightarrow{\theta_5}$y $\overrightarrow{G}$, y a hacer la conversión valor por valor a través de la función:

```
function [arreglo_salida] =
comunicar(v1,v2,v3,v4,v5,v6,matrizorden,bytepa,bytept,k)
```

Los parámetros `v1,v2,v3,v4,v5 y v6` corresponden a los $\overrightarrow{\theta_{1i}}$, $\overrightarrow{\theta_{2i}}$, $\overrightarrow{\theta_{3i}}$, $\overrightarrow{\theta_{4i}}$, $\overrightarrow{\theta_{5i}}$y $\overrightarrow{G_i}$, respectivamente; la `matrizorden,` es la matriz paralela a los vectores anteriores que contiene el orden que se deben mover los ángulos explicado en la Sección 5.7.4; `bytepa y bytept` son los bytes correspondientes a los números

de paquetes, y `k` es el valor de inicio para que se vayan añadiendo los bytes de tres en tres e ir considerando las posiciones para agregar el siguiente y formar el arreglo de salida (ver Apéndice 7.6).

5.9. Implementación de la librería HIDAPI para la comunicación

Para poder transmitir al PIC los datos contenidos en el arreglo de salida descrito en la Sección 5.8, se utiliza la librería HID API mencionada en la Sección 2.5.6 la cual puede ser manipulada por MATLAB de forma sencilla. El primer paso al descargar los archivos propios de la HIDAPI de [18] es hacer la compilación del archivo hidapi.sln; para efectos de este trabajo de grado se utiliza a la herramiento Microsoft Visual Studio Express 2013 para escritorio de Windows, el cual ofrece la opción para compilar en x32 y x64. Una vez que se realiza la compilación, se genera la librería hidapi.dll, y para utilizar hidapi.dll desde MATLAB se debe hacer uso de lo archivos MEX (ver Seccion 2.8.2.2), ademas de ubicarla dentro de la carpeta que contenga el archivo .m de MATLAB. Para generar el archivo MEX se necesita un compilador soportado por MATLAB en este caso se utiliza Microsoft Visual Studio 2010. Para hacer la configuración del compilador se ejecuta la instrucción `mex -setup` en la ventana de comandos de MATLAB y se siguen las instrucciones. Hecho esto, es posible hacer uso de la librería hidapi y de acuerdo al algoritmo próximamente descrito lograr una comunicación exitosa (ver Figura 5.26).

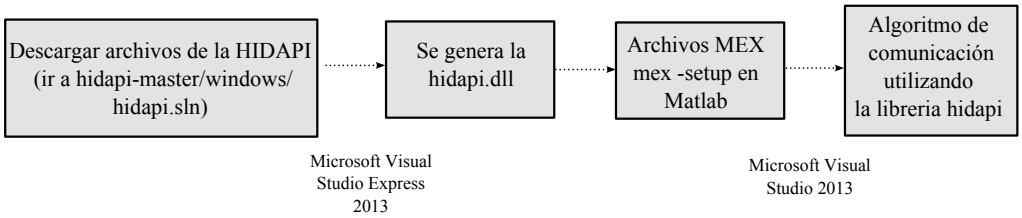


Figura 5.26: Funcionamiento de la librería HIDAPI para ser usada con MATLAB.

El algoritmo para el envío de paquetes de datos es el siguiente:

`loadlibrary ('hidapi','hidapi.h');` Esta instrucción se encarga de cargar la librería para ser utilizada.

`Ptr_Disp=`

`calllib('hidapi','hid_open',hex2dec('1235'),hex2dec('0002'),`

`libpointer('voidPtr'));` Abre el canal de comunicación para enviar los datos.

`buffer_salida=arreglo_salida;` Define el buffer de salida que contiene los datos a enviar.

`Ptr_buffer_salida=libpointer('uint8Ptr',buffer_salida);` Crea el objeto tipo puntero con los datos de salida.

`calllib('hidapi','hid_write',Ptr_Disp,Ptr_buffer_salida,65);` Recibe el objeto de tipo puntero, y se encarga de enviar los datos al PIC.

Para la recepción de datos, que sirve para la verificación de una correcta comunicación se utilizan las siguientes instrucciones:

`buffer_entrada=zeros(1,64);` Define al buffer de entrada donde se almacenan los datos provenientes del PIC.

`calllib('hidapi','hid_read_timeout',Ptr_Disp,Ptr_buffer_entrada,`

`64,2000);` Recibe el puntero del dispositivo y lee los datos recibidos.

`buffer_entrada=get(Ptr_buffer_entrada,'Value');` Obtiene el valor en el puntero del buffer de entrada.

`calllib('hidapi','hid_close',Ptr_Disp);` Termina la comunicación con el dispositivo.

Este proceso es realizado tantas veces como paquetes lleguen a existir para definir las trayectorias del Rhino XR-3, como lo muestra la Figura 5.27.

5.10. Roboworks

Para el modelo tridimensional del Rhino utilizado se ajustaron las medidas y ciertas características del modelo propuesto en [2]. La animación permite observar el movimiento por pasos de las cinco articulaciones y también la apertura y cierre del gripper. El archivo de datos debe tener una estructura especifica para poder ser reconocido por Roboworks. En la Figura 5.28 se muestra la dinámica utilizada.

En cuanto a la estructura del archivo de datos generado por la interfaz, este debe poseer un encabezado definido por las etiquetas dadas a las articulaciones

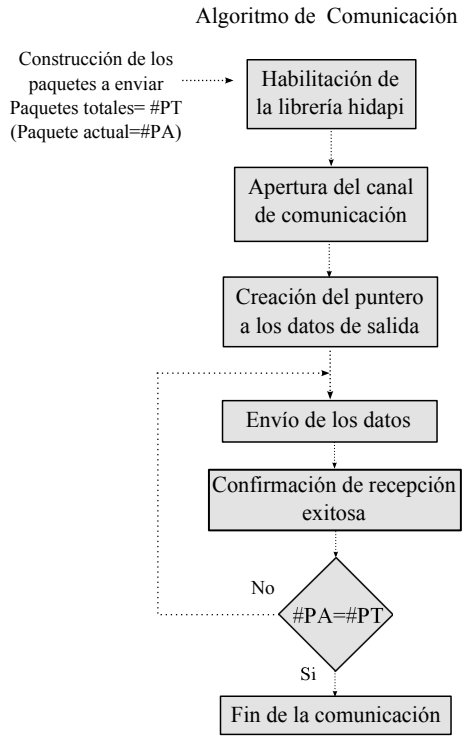


Figura 5.27: Esquema de funcionamiento del algoritmo de comunicación.

en Roboworks así como se muestra en la Figura 5.29. Además para ver el movimiento real del robot, cada linea del archivo es diseñada para que se observe el cambio progresivo en una articulación por vez, cambiando una y manteniendo las otras iguales, hasta que llegue al punto final y pueda pasar a la siguiente articulación, de acuerdo al orden de movimiento determinado con el algoritmo explicado en la Sección 5.7.4.

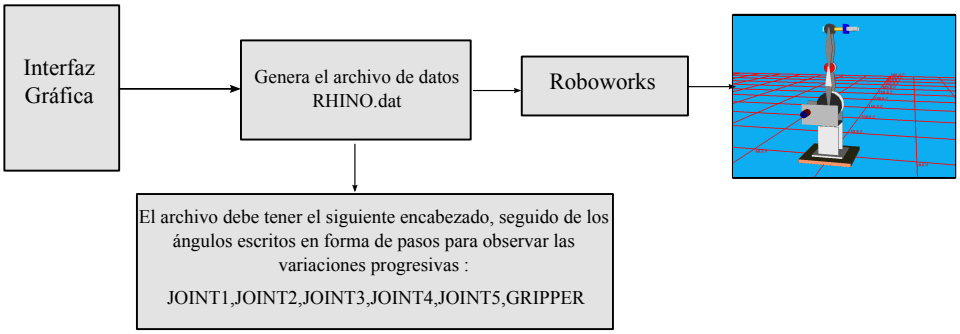


Figura 5.28: Dinámica de funcionamiento entre la Interfaz Gráfica y Roboworks.

```
Joint1 Joint2 Joint3 Joint4 Joint5 Gripper
0.00 90.00 0.00 90.00 0.00 2.50
0.00 90.00 0.00 88.66 0.00 2.50
0.00 90.00 0.00 87.32 0.00 2.50
0.00 90.00 0.00 85.97 0.00 2.50
0.00 90.00 0.00 84.63 0.00 2.50
0.00 90.00 0.00 83.29 0.00 2.50
0.00 90.00 0.00 81.95 0.00 2.50
0.00 90.00 0.00 80.61 0.00 2.50
0.00 90.00 0.00 79.27 0.00 2.50
0.00 90.00 0.00 77.92 0.00 2.50
0.00 90.00 0.00 76.58 0.00 2.50
0.00 90.00 0.00 75.24 0.00 2.50
0.00 90.00 0.00 73.90 0.00 2.50
0.00 90.00 0.00 72.56 0.00 2.50
0.00 90.00 0.00 71.22 0.00 2.50
0.00 90.00 0.00 69.87 0.00 2.50
0.00 90.00 0.00 68.53 0.00 2.50
0.00 90.00 0.00 67.19 0.00 2.50
0.00 90.00 0.00 65.85 0.00 2.50
```

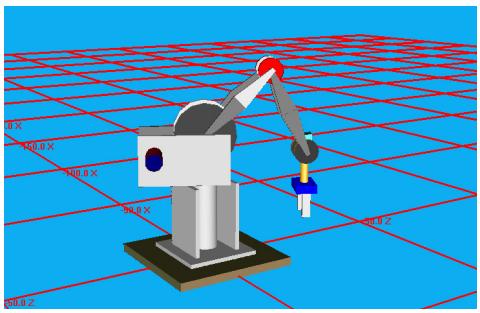


Figura 5.29: Estructura del archivo de datos leído por Roboworks,

6 Conclusiones y Recomendaciones

Una vez probado y calibrado el sistema y la unidad de control del brazo antropomórfico Rhino XR-3, se procede a exponer las conclusiones y recomendaciones pertinentes.

6.1. Conclusiones

En el presente trabajo se logró construir el circuito y la interfaz de control por computador que constituyen la Unidad de Control Cinemático para el autómata antropomórfico XR-3 de cinco grados de libertad del Laboratorio de Robótica y Visión Industrial, el cual puede ser controlado a través del movimiento pleno de sus articulaciones para la realización de tareas programadas que actúen sobre un objeto de determinado color empleando visión por computador a través de cámara web.

Se diseñó el circuito electrónico para el control de motores, en donde se hizo uso de componentes como el PIC18F2550, integrados puente L6203b, los cuales junto a la lectura de un encoder lograron garantizar la posición angular correcta, con un error despreciable, obteniéndose así los mejores resultados durante las pruebas.

Fue realizada la construcción de circuitos impresos para el control de motores y comunicación USB, prefiriéndose una implementación en módulos interconectados que compactan el diseño y facilita reparaciones. Se incorporó una pantalla LCD para validar la data recibida por USB y un Bootloader que permite a la unidad de control ser reprogramada desde el mencionado bus sin necesidad de abrir la caja que contiene al circuito.

Mediante un estudio matemático exhaustivo se desarrolló un modelo capaz de definir el comportamiento del Rhino XR-3 tanto en la Cinemática Directa como Inversa, basados en el planteamiento de ecuaciones que incluyen las

consideraciones del área de trabajo del robot presentes en el Laboratorio de Robótica y Visión Industrial.

Se desarrolló un modelo empírico para determinar el orden de movimiento secuencial de las articulaciones con el cual el brazo antropomorfico debe moverse, y que garantiza un adecuado desplazamiento de las partes del robot sin atentar contra su integridad.

El modelo cinemático inverso fue evaluado a través de dos formas, una de ellas mediante un algoritmo diseñado para visualizar el correcto posicionamiento del robot al momento de dirigirse a un punto especifico del plano de trabajo, utilizando el modelado 2D y 3D que ofrece Matlab, y la otra utilizando el modelador tridimensional Roboworks, que permitió también evaluar no solo el modelo cinemático inverso si no también el directo.

La implementación del código descrito en la Sección 5.3.1 que permite generar una animación en tiempo real gracias a la presentación sucesiva de imágenes que van mostrando los cambios de posición del robot, funcionó de forma satisfactoria para poder verificar en pleno desarrollo del trabajo de grado la veracidad del modelo matemático expuesto en la Sección 5.2, ya que mediante el modelador Roboworks, sólo era posible comprobar la veracidad del planteamiento al final del desarrollo completo del trabajo debido a su rígida estructura de manejo.

Se logró la discriminación de objetos de determinado color en una imagen del plano de trabajo del robot, adquirida mediante una cámara web, donde el algoritmo capaz de realizar dicha discriminación es lo bastante robusto para lidiar con factores externos como sombras, gracias al uso de parámetros que permiten ajustar el código a distintos escenarios donde el Rhino XR-3 puede trabajar.

La utilización del algoritmo de procesamiento de imágenes diseñado, expuesto en la Sección 5.6.1, arrojó los resultados esperados a pesar de la poca iluminación y la generación de gran cantidad de sombras y perturbaciones presentes en el Laboratorio de Robótica y Visión Industrial.

Finalmente se desarrolló la interfaz gráfica que permite gestionar tareas a través de la visión por computador, ofreciendo una gran variedad de alternativas para realizar el control cinemático directo e inverso. Este ultimo puede darse mediante tareas automáticas utilizando el procesamiento de imágenes, o a través de tareas manuales especificadas directamente por el usuario por medio de clicks o por el ingreso directo de los valores que permitan lograr el control

deseado. Además, la interfaz contiene su respectiva área de visualización de resultados y la opción de ejecutar la simulación y posterior comunicación USB con el circuito de control del Rhino XR-3.

6.2. Recomendaciones

Se plantean las siguientes recomendaciones y sugerencias:

Para el diseño de circuitos para el control de motores desde microcontrolador:

- Aislar completamente las señales y alimentación del microcontrolador empleando opto-acopladores, usar fuentes lineales para la alimentación de motores, emplear de ser posible circuitos integrados puente con todas sus respectivas protecciones capacitivas y diodos de protección contra corrientes inversas.
- Elaborar un Teach Pendat ó programador manual, que permita la programación de tareas sin necesidad de un computador, para ello solo deben enviarse datos al puerto USB en el orden y tiempo adecuado.
- Optimizar los lazos de control de los motores. Para ello puede crearse una siguiente versión de unidad de control que lea un segundo encoder para estimar la velocidad, pudiendo incorporar también un multiplexor analógico terminado en amperímetros pasivos conectados por opto-acoplador al convertidor A/D del microcontrolador para medir también la corriente de carga de los motores.

Para el software de control:

- Mejorar el funcionamiento del algoritmo que discrimina un objeto, complementándolo con la posibilidad de identificar más de un objeto del mismo color, involucrando así la ejecución de varias tareas de desplazamiento de objetos de forma consecutiva y automática.
- Ampliar la gama de colores que el algoritmo es capaz de discriminar los cuales son solo rojo, verde y azul, utilizando los factores y parámetros que juegan con la intensidad de los colores admitidos.
- Añadir otras formas de figuras con las que el Rhino XR-3 pueda trabajar, distintas a paralelepípedos, cubos o cilindros, manipulando el algoritmo que determina la correcta orientación de la herramienta del robot y adaptándolo para otra variedad de formas.

- Existe la alternativa de cambiar la consideración de que la herramienta de trabajo siempre se encuentre perpendicular al plano Z, adaptando al robot a posicionarse en otros puntos del espacio sin importar al plano al que pertenezcan.

7 Apéndices

7.1. Apéndice A: Algoritmo de procesamiento de la imagen

```
function [px,py,dx,dy,jx,jy,dmin] =
coordenadas(img,anchoi,largoi,anchop,largop,color,tol,mifactor)
%color 'r' 'R' 'g' 'G' 'b' 'B'
%tol valores por defecto 0.4 0.2
%mi factor: mayor que 1 1.2 1.6.


%Matriz de 3 dimensiones
[row col plane] = size(img)
%Convirtiendo la matriz de enteros en valores double
img = double(img);
%Matriz de ceros donde se copiara la matriz final
C = zeros(row,col,plane);
if plane ~= 3
    disp('Input should be a color image');
    return;
end
%Factor: maximo valor de la matriz multiplicado por un valor de tolerancia
%que servira para poder determinar si se esta en un color especifico
factor = max(img(:)) * tol;

switch color
    case {'R','r'}
        for i = 1:row
            for j = 1:col
                if (img(i,j,1) > factor && img(i,j,1) == max([img(i,j,1)
img(i,j,2)*mifactor img(i,j,3)*mifactor]))
                 %Se mantiene el valor 0 indicando negro
                else
                    C(i,j,1:3) = 255;
                end
            end
        end
    case {'G','g'}
        for i = 1:row
            for j = 1:col
                if (img(i,j,2) > factor && img(i,j,2) ==
max([img(i,j,1)*mifactor img(i,j,2) img(i,j,3)*mifactor]))
                    %Se mantiene el valor 0 indicando negro
                else


C(i,j,1) = 255;
                    C(i,j,2) = 255;
                    C(i,j,3) = 255;
                end
            end
        end
    case {'B','b'}
        for i = 1:row
            for j = 1:col
                if (img(i,j,3) > factor && img(i,j,3) ==
max([img(i,j,1)*mifactor img(i,j,2)*mifactor img(i,j,3)]))
                %Se mantiene el valor 0 indicando negro
                else
```

```
                    C(i,j,1) = 255;
                    C(i,j,2) = 255;
                    C(i,j,3) = 255;
                end
            end
        end
    otherwise
        disp('unknown method');
end

BW4 = im2bw(imcomplement(C));
% imshow(BW4)
BW5 = imfill(BW4,'holes');

%Acumulador que guarda los elementos de la matriz que pertenecen al objeto
acumulador = zeros(2,1);
cont = 1; %cuenta la cantidad de pixeles que pertenecen al objeto

for i = 1:row
            for j = 1:col

                if BW5(i,j)==1
                    acumulador(1) = acumulador(1) + i;
                    acumulador(2) = acumulador(2) + j;
                    cont = cont + 1;
                end

            end
end
centro = acumulador/cont;
b=0;
for i = 1:row
            for j = 1:col
                    if BW5(i,j)==0
                        d=sqrt((i-centro(1))^2+(j-centro(2))^2);

                        if b==0
                            dmin=d;
                            xmin=i;
                            ymin=j;
                            b=1;
                        elseif d < dmin
                            dmin=d;
                            xmin=i;
                            ymin=j;
                        end

                    end
            end
end


% jx = ymin;Punto con distancia mas corta a partir del centro
% jy = xmin;
```

```
jx = ((ymin) - (largoi/2))*(largop/largoi)
jy = abs((anchoi -(xmin))*anchop/anchoi)


figure,imshow(uint8(img),[]);
figure,imshow(uint8(BW5),[]);
impixelinfo
hold on;
plot(centro(2), centro(1), '+');
plot(ymin, xmin, 'r*');
%
% dy=centro(1)
% dx=centro(2)
%
% px = (dx - (largoi/2))*(largop/largoi)%Centro del objeto en cm
% py = abs((anchoi -dy)*anchop/anchoi)

py=centro(1)
px=centro(2)

dx = (px -
(largoi/2))*(largop/largoi)%Centro
del objeto
dy = abs((anchoi -
py)*anchop/anchoi)


end
```

7.2. Apéndice B: Algoritmo que calculo de θ_5 a partir del procesamiento de imágenes

```
function [ theta5 ] = theta5nuevo(dx,dy,jx,jy,theta1g)
dx
dy
jx
jy
theta1g
theta1=(theta1g*pi/180);
modulodj=sqrt((jx-dx)^2+(jy-dy)^2);
djx=(jx-dx)/modulodj
djy=(jy-dy)/modulodj

if djy>0 && djx>0
    theta5r= theta1+ atan(djx/djy)
elseif djy<0 && djx<0
    theta5r= theta1+(atan(abs(djx/djy))-pi)
elseif djy>0 && djx<0
    theta5r= theta1+(-atan(abs(djx/djy)))+pi
elseif djy<0 && djx>0
    theta5r= theta1 +(-atan(abs(djx/djy)))
end

theta5= ((theta5r*180/pi))
end
```

7.3. Apéndice C: Algoritmo que calcula la Cinemática Inversa

```
function [theta1,theta2,theta3,theta4] = calculothetas(x,y,zz)

global R;
global z;
z = zz;
R = sqrt(x^2 + y^2);
% hold off
guess = [0; 0];
[sol, fval] = fsolve(@mifunc, guess);
if (x<0)
    theta1=pi -abs(atan(y/x));
elseif (x==0) & (y==0)
    theta1=0;
else
    theta1= atan(y/x);
end
theta2 = sol(1);
% % Para que me de normal
theta3 = (-1)*sol(2);
theta4 = pi/2-(pi - theta2 + theta3-pi);
```

```
function [vt1 vt2 vt3 vt4] =
Vectorthetas(vx,vy,vz)

t=size(vx);

vt1 = zeros(t(1),1);
vt2 = zeros(t(1),1);
vt3 = zeros(t(1),1);
vt4 = zeros(t(1),1);


for i=1:t(2)

[theta1fr,theta2fr,theta3fr,theta4f
r]=calculothetas(vx(i),vy(i),vz(i))
;

theta1f= (theta1fr*180/pi);
theta2f= (theta2fr*180/pi);
theta3f= (theta3fr*180/pi);
theta4f= (theta4fr*180/pi);

vt1(i)=theta1f;
vt2(i)=theta2f;
vt3(i)=theta3f;
vt4(i)=theta4f;

end

end
```

7.4. Apéndice D: Algoritmo para verificar funcionamiento de la Cinemática Inversa

```
function [theta1fg,theta2fg,theta3fg,theta4fg] =
animacion(theta1ig,theta2ig,theta3ig,theta4ig,x,y,zz)
[theta1f,theta2f,theta3f,theta4f]=calculothetas(x,y,zz);


theta1i=(theta1ig*pi/180);
theta2i=(theta2ig*pi/180);
theta3i=(theta3ig*pi/180);
theta4i=(theta4ig*pi/180);


theta1fg= (theta1f*180/pi);
theta2fg= (theta2f*180/pi);
theta3fg= (theta3f*180/pi);
theta4fg= (theta4f*180/pi);

nd=200;
d1=(theta1f-theta1i)/nd;
d2=(theta2f-theta2i)/nd;
d3=(theta3f-theta3i)/nd;
d4=(theta4f-theta4i)/nd;

for n=1:1:nd

    t1m=theta1i+n*d1;
    t2m=theta2i+n*d2;
    t3m=theta3i+n*d3;
    t4m=theta4i+n*d4;
    graficart(t1m,t2m,t3m,t4m);
    grid on;
    plot3(x,y,0,'r*');
    pause(0.00005)
    hold off

end

end
```

7.5. Apéndice E: Algoritmo que genera el orden de movimiento de las articulaciones

```
function [M,o] = tipomov(xi,yi,zi,xf,yf,zf,vt5i,vgripperi,vt5f,vgripperf,M)
ri=sqrt(xi^2+yi^2);
rf=sqrt(xf^2+yf^2);

 vx=[xi xf ];
 vy=[yi yf ];
 vz=[zi zf];
 [vt1 vt2 vt3 vt4] = Vectorthetas(vx,vy,vz);

 i=1;
if (rf<ri) %se acerca

     oacerca=[5;4;2;1;3;6];
     o=oacerca;

res=generardatf(vt1(i+1),vt2(i+1),vt3(i+1),vt4(i+1),vt5f,vgripperf,vt1(i),vt2
(i),vt3(i),vt4(i),vt5i,vgripperi,oacerca);
        M=[M;res];

elseif (rf>ri) %se aleja
oaleja=[1;3;2;4;5;6];
 o=oaleja;

res=generardatf(vt1(i+1),vt2(i+1),vt3(i+1),vt4(i+1),vt5f,vgripperf,vt1(i),vt2
(i),vt3(i),vt4(i),vt5i,vgripperi,oaleja);
        M=[M;res];

elseif (rf==ri)

 if (zi>zf)%baja

     obaja=[5;4;3;1;2;6];
      o=obaja;


res=generardatf(vt1(i+1),vt2(i+1),vt3(i+1),vt4(i+1),vt5f,vgripperf,vt1(i),vt2
(i),vt3(i),vt4(i),vt5i,vgripperi,obaja);
        M=[M;res];


    else %sube

  osube=[2;1;3;4;5;6];
  o=osube;


res=generardatf(vt1(i+1),vt2(i+1),vt3(i+1),vt4(i+1),vt5f,vgripperf,vt1(i),vt2
(i),vt3(i),vt4(i),vt5i,vgripperi,osube);
        M=[M;res];


end

end
```

7.6. Apéndice F: Algoritmo que crea el arreglo de salida para la comunicación

```
function [arreglo_salida] =
comunicar(v1,v2,v3,v4,v5,v6,matrizo
rden,bytepaqa,bytepaqt,k)
arreglo_salida=zeros(1,65);
tamano=size(v1);
marca=4;
arreglo_salida(1,1)=1;
arreglo_salida(1,2)=bytepaqa;
arreglo_salida(1,3)=bytepaqt;
pasos=2;


    if (mod((tamano(2)-1),3)==0) &
(k==tamano(2))
        pasos=0;
    elseif (mod((tamano(2)-2),3)==0
&(k+1==tamano(2)))
        pasos=1;
    end



 for i=k:k+pasos

    for j=1:6

    if matrizorden(i,j)==1


[b1,b2,b3]=paquete(v1(i),1);
        arreglo_salida(1,marca)=b1;

arreglo_salida(1,marca+1)=b2;

arreglo_salida(1,marca+2)=b3;
        marca=marca+3;
    elseif matrizorden(i,j)==2


[b1,b2,b3]=paquete(v2(i),2);
        arreglo_salida(1,marca)=b1;

arreglo_salida(1,marca+1)=b2;

arreglo_salida(1,marca+2)=b3;
        marca=marca+3;

    elseif matrizorden(i,j)==3


[b1,b2,b3]=paquete(v3(i),3);
        arreglo_salida(1,marca)=b1;

arreglo salida(1,marca+1)=b2;

arreglo_salida(1,marca+2)=b3;
        marca=marca+3;
    elseif matrizorden(i,j)==4


[b1,b2,b3]=paquete(v4(i),4);
        arreglo_salida(1,marca)=b1;

arreglo_salida(1,marca+1)=b2;

arreglo_salida(1,marca+2)=b3;
        marca=marca+3;
    elseif matrizorden(i,j)==5


[b1,b2,b3]=paquete(v5(i),5);
        arreglo_salida(1,marca)=b1;

arreglo_salida(1,marca+1)=b2;

arreglo_salida(1,marca+2)=b3;
        marca=marca+3;
    elseif matrizorden(i,j)==6


[b1,b2,b3]=paquete(v6(i),6);
        arreglo_salida(1,marca)=b1;

arreglo_salida(1,marca+1)=b2;

arreglo_salida(1,marca+2)=b3;
        marca=marca+3;
    end
    end
 end


marca;
end
```

7.6.0.1. Algoritmos que convierten un angulo y un identificador de motor en tres bytes.

```
function [byte1,byte2,byte3 ] = paquete(angulo,motor)
if angulo>0
    signo=0;
else
    signo=1;
end
% if bandera==1
%     bit1=1;
% else
%     bit1=0;
% end
if motor==1
    bytedec=[0 0 0 0 signo 0 0 1 ];

elseif motor==2
     bytedec=[0 0 0 0 signo 0 1 0 ];

elseif motor==3
 bytedec=[0 0 0 0 signo 0 1 1 ];

elseif motor==4
    bytedec=[0 0 0 0 signo 1 0 0 ];

elseif motor==5
    bytedec=[0 0 0 0 signo 1 0 1 ];

elseif motor==6
    bytedec=[0 0 0 0 signo 1 1 0];

end
bytedec;
cadenastr='';
for i=1:8
cadenastr=[cadenastr num2str(bytedec(1,i))];
end
c=cadenastr;
byte1=base2dec(c,2);

[byte2,byte3]=conversion(angulo);
%
% byte1
% byte2
% byte3

end
```

```
function [byte1,byte2] = conversion(angulo)
%Inclusion del valor decimal en dato a enviar
n=abs(angulo*10);
%Conversion de decimal a binario
valor=dec2bin(n);
t=size(valor);
cadenadec=zeros(1,12);
a=num2str(valor);
cadenastr='';
if t(2)<12
for i=1:t(2)

    ad=base2dec(a(i),10);
    cadenadec(1,12-t(2)+i)=ad;

end
for i=1:12
cadenastr=[cadenastr num2str(cadenadec(1,i))];
end
cadenastr;
else
   cadenastr=a;
end
c=cadenastr;
%Descomposicion en dos bytes
s=strcat(c(5),c(6),c(7),c(8),c(9),c(10),c(11),c(12));
s1=strcat(0,0,0,0,c(1),c(2),c(3),c(4));
%Byte 1 en hex
byte1=base2dec(s,2);
% Byte 2 en hex
byte2=base2dec(s1,2);
res=bin2dec(s);
res1=bin2dec(s1);

end
```

Bibliografía

[1] Sanz Fernández, Wilmer. (2009). *Cinemática de robots industriales*. Universidad de Carabobo.

[2] Colina, Fausto. (2005). *Diseño e implementación de una interfaz controladora para manipulador robotico RHINO XR3 por comunicación paralela*. Universidad de Carabobo.

[3] Soares L. y Casanova V. (2006). *An Educational Robotic Workstation based on the Rhino XR4 robot*. 36th ASEE/IEEE Frontiers in Education Conference.

[4] Ollero, Anibal. (2001). *Robótica, manipuladores y robots móviles*. Editorial Marcombo S.A.

[5] Rhino Robotics Ltd. Model xr-3. [En línea] http://www.rhinorobotics.com/xr3 flyer.html [U. consulta:01/02/15].

[6] Trujillo D. y Frassato A. (1995). *Controlador de Brazo Robótico*. Universidad de Carabobo.

[7] Usategui J. y Romero S. *Diseño práctico de aplicaciones 2da Parte*. Mc Graw Hill, (2009).

[8] Blanchard, Eugine. (2001). The using mosfets website. h-bridge using p and n channel fets. [En línea] http://www.armory.com/rstevew/Public/Motors/H-Bridges/Blanchard/figure-1.htm [Última consulta: 6/01/15].

[9] Floyd, Thomas. (2004). *Fundamentos de Sistemas Digitales 7ma edición*. Pearson Prentice Hall.

[10] Philips Ecg Inc. (1987). *ECG Semiconductors Master Replacement Guide*.

[11] Frassato L. y Gonzalez A. (2014). *Construcción de un tomógrafo por impedancia eléctrica para imagenología médica.* Universidad de Carabobo.

[12] Ogata, Katsuhiko. (2003). *Ingeniería de Control Moderno*. Pearson.

[13] Microchip Technology Inc. (2009). *PIC18F2455/2550/4455/4550 Data Sheet*. MICROCHIP.

[14] Clavijo, Juan. (2011). *Diseño y simulación de sistemas microcontrolados en lenguaje C*.

[15] Axelson, Jan. (2001). Usb complete. [En línea] http://guabiruba.sc.gov.br/EmbeddedProgram/USB20Complete.PDF [Última consulta: 6/01/15].

[16] Ordoñez, Daniel. (2012). Transferencia de datos vía usb con matlab. En línea] http://colab-matlab.blogspot.com/2012/03/transferencia-de-datos-via-usb-con.html [Última consulta: 10/11/14].

[17] Creative Commons. (2013). Comunicación usb con el pic pic18f4550. [En línea] http://www.aquihayapuntes.com/indice-practicas-pic-en-c/comunicacion-usb-pic18f4550-utilizando-la-clase-cdc.html [Última consulta: 12/11/14].

[18] Ott A. (2014). Hidapi github. [En línea] https://github.com/signal11/hidapi. [Última consulta: 4/12/14].

[19] Otero, Juan. (2002). *Modelo interpolativo punto-superficie para calbrado de cámaras en Visión Artificial*. PhD thesis, Universidad Politécnica de Madrid.

[20] Rojas T. y Sanz W. (2008). *Sistema de visión por computadora para la detección de objetos esféricos a través de la transformada de Hough*. Revista Ingeniería UC. Vol. 15, No 1, 77-87.

[21] Sanz Fernández, Wilmer. (2003). *Diapositivas de Visión por Computadora*. Universidad de.

[22] Esqueda J. y Pelafox L. (2005). *Fundamentos de procesamiento de imágenes*. Universidad Autónoma de Baja California.

[23] De Miguel Anasagasti ,Pedro. (1996). *Computadores paralelos y evaluación de prestaciones*. Colección Ciencia y Técnica.

[24] Mathworks. Matlab help. [En línea] http://es.mathworks.com/help/matlab/ [Ú. consulta: 20/11/14].

[25] Puglesi A. y Marcucchi G. (2013). Comunicación usb para aplicaciones de control con matlab/simulink y placa esd-32u. *14 th Argentine Symposium on Technology, AST*.

[26] Barragan, Diego. (2009). Manual de interfaz gráfica de usuario en matlab. [En línea] https://www.dspace.espol.edu.ec/bitstream/123456789/10740/11/MA TLABGUIDE.pdf [Última consulta: 6/01/15].

[27] Newtonium. Roboworks. [En línea] http://www.newtonium.com/publichtml/Products/RoboWorks/RoboWorks.htm [Última consulta 3/02/15].

[28] Barrios, Maritza. (1998). *Manual de trabajos de grado, de especialización y maestría y tesis doctorales.* Universidad Pedagógica Experimental Libertador.

[29] Ilis, Alfonzo. (1994). *Técnicas de investigación bibliográfica.* Contexto Ediciones.

[30] Rojas, Teddy. (2008). *Diseño de un sistema de visión artificial para un manipulador industrial antropomórfico.* Universidad de Carabobo.

[31] Castañeda A. y Adrian E. (2011). *Implementación de un sistema de manipulación y control mediante visión artificial para un autómata industrial SCARA.* Universidad de Carabobo.

[32] Azor, Ruben. (2011). *Herramientas computacionales en ciencias exactas.* Universidad de Mendoza. Facultad de Ingenieria.

[33] Reyes, Fernando. (2012). *Matlab aplicado a la robótica y mecatrónica.* Editorial Alfaomega.

[34] Franco A. y Rodriguez C. (2005). *Modelación y Animación con el Roboworks versión 2.0.* Universidad EAFIT.