UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA: INGENIERÍA ELECTRÓNICA

Tesis previa a la obtención del título de: INGENIERO ELECTRÓNICO

TEMA:
" Diseño y Construcción de un módulo manipulador de objetos para el Robot móvil Robotino de Festo para la Carrera de Ingeniería Electrónica, Sede Quito Campus Sur"

AUTOR:
CRISTIAN CRISTOBAL CUJI CUJI

DIRECTOR:
RODRIGO TUFIÑO CÁRDENAS

Quito, junio 2013

DECLARATORIA DE RESPONSABILIDAD Y AUTORIZACIÓN DE USO DELTRABAJO DE GRADO

Yo Cristian Cristóbal Cuji Cuji autorizo a la Universidad Politécnica Salesiana la publicación total o parcial de este trabajo de grado y su reproducción sin fines de lucro.

Además declaro que los conceptos y análisis desarrollados y las conclusiones del presente trabajo son de exclusiva responsabilidad del autor.

_____

Cristian Cristobal Cuji Cuji
CI: 171800222-1

DEDICATORIA

Dedicado a mi linda familia quienes me apoyaron incondicionalmente durante mis estudios universitarios: Anita, Pedrito, David, Rodrigo, Carmita, Alex, Almudena y a mi preciosa Maricel.

AGRADECIMIENTOS

A la Universidad Politécnica Salesiana, por permitirme recibir una educación de calidad con profesores de grandes conocimientos y experiencia. En especial quiero agradecer al Ing. Rodrigo Tufiño, por toda la confianza, apoyo y paciencia recibida durante el desarrollo del proyecto de tesis, a la Ing. Johanna Celi y al Ing. Carlos Pillajo por las observaciones realizadas al proyecto.

Y en general a todos los profesores que tuve la suerte de conocer en mi paso por la universidad y finalmente a una gran profesor Ing. Germán Arévalo.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Para empezar con el proyecto el estudio se enfoco de los principales temas del proyecto: Robótica, Electrónica y Mecánica. Se describió un resumen de la historia, características y situación actual de la tecnología, además del uso de conocimientos y herramientas que se van a ser utilizadas en el desarrollo del proyecto.

Durante el Diseño, Desarrollo e Implementación se explica y detalla la elaboración del proyecto en sus tres áreas: La construcción del sistema Mecánico, La construcción del sistema de control electrónico, y La implementación del sistema driver de control.

Durante el diseño y desarrollo se presenta datos técnicos, medidas y características del sistema mecánico. Pero al no ser el tema principal de estudio no se presenta mayores detalles de la elaboración del módulo. Lo más importante es obtener un sistema mecánico que pueda ser controlado con un sistema electrónico.

Se presenta con mayor profundidad información de características, funcionamientos, planos esquemáticos, algoritmos de programación y todo lo correspondiente al sistema electrónico de control. Se dedico mayor énfasis en explicar el funcionamiento y los diferentes elementos electrónicos presentes en la tarjeta de control.

Finalmente se describe la implementación de los dos sistemas anteriores; Mecánico y Electrónico. En un dispositivo driver de control desarrollado con la herramienta RobotinoView_API_v2.8.4 que es casi la última versión en ser publicada por FESTO. Este subcapítulo se presenta a mi forma personal el mayor aporte de mi trabajo de investigación, ya que las herramientas API son poco conocidas pero poseen un amplio campo para aplicaciones y desarrollo de nuevas soluciones.

ABSTRACT

To begin the project, the study was focused on the main themes of the project: Robotics, Electronics and Mechanics. It described an overview of the history, characteristics and current situation of technology, and the use of knowledge and tools that will be used in the development of the project.

During the design, development and implementation is explained and detailed the development of the project in three areas: Mechanical System Construction, Building of electronic control system, and implementation of driver control system.

During the design and development presents technical data, dimensions and characteristics of the mechanical system. But not being the main subject of study is not presented further details of the development of the module. Most important is obtaining a mechanical system which can be controlled with an electronic system.

It presents with further information of features, performances, schematics, programming algorithms and everything for the control electronics. Greater emphasis is devoted to explain the operation and the various electronic elements present in the control card.

Finally this describes the implementation of the two systems, Mechanical and Electronic. In a control device driver tool developed RobotinoView_API_v2.8.4 which is almost the final version to be published by FESTO. This subchapter is presented to me personally the greatest contribution of my research, because the tools are not well known API but have a wide scope for developing new applications and solutions

INTRODUCCIÓN

FESTO © y FESTO DIDACTIC son compañías de origen Alemán, que se dedican al desarrollo de Sistemas de Automatización y Control Industrial. El Robot Móvil ROBOTINO es uno de sus productos desarrollados exclusivamente para generar investigación y educación de estudiantes. Robotino es un Sistema Robot Móvil Autónomo, está constituido por un computador que trabaja en sistema operativo Linux, también cuenta con una base omnidireccional, sensor de choque, sensores infrarrojos, sensores de distancia ultrasónicos y una cámara en color VGA. El diseño de Robotino es modular, y puede ser fácilmente equipado con una variedad de accesorios, gracias a un módulo de entradas y salidas.

Utilizando dispositivos electrónicos y herramientas propias del Robotino, se busca desarrollar un Sistema Manipulador de Objetos, tomando como base una tarjeta de control y adquisición de datos.

Con el desarrollo del nuevo módulo manipulador el Sistema Robot Móvil Robotino se convierte en un Sistema de Transporte Guiado Autónomo, término acuñado en ingles como (AGV Automated Guided Vehicle) por la misma empresa FESTO, con la característica particular de brindar un grado de libertad adicional al funcionamiento del Robotino.

Para el desarrollo del proyecto los lineamientos son los siguientes: Diseño Mecánico, Diseño Electrónico / Eléctrico, Desarrollo del Software de control e Implementación en el Robotino.

Aunque la construcción del sistema mecánico en su mayoría será realizada por un mecánico industrial, el diseño de la estructura fue diseñado previamente en AutoCAD. Tomando en cuenta medidas, ubicación de elementos y transmisión de movimiento. Luego se debe desarrollar un sistema electrónico compatible y que permita poner en funcionamiento el sistema mecánico y junto con el desarrollo del sistema electrónico también el desarrollo del software de control, programación de dispositivos e

implementación en el Robotino por medio de un lenguaje de programación propio conocido como ROBOTINO VIEW.

Para el desarrollo del proyecto se plantean los siguientes capítulos:

1. **Hipótesis y Tesis**.- Introducción al tema, Alcances, Planteamiento de los objetivos y problema. Son entre otros los temas desarrollados.
2. **Estado del Arte**.- Es una recopilación o marco teórico de los conocimientos necesarios para el desarrollo del proyecto, como por ejemplo: Robótica, Mecánica, Sistema de Control Electrónico, Sensores y Actuadores entre otros temas de interés para el proyecto.
3. **Desarrollo e Implementación**.- Es la descripción del funcionamiento, desarrollo e implementación del trabajo realizado en las áreas de conocimientos antes mencionados, como por ejemplo: Construcción del Sistema Mecánico, Desarrollo de la Tarjeta de control, Desarrollo de la programación de los dispositivos y la Implementación en el Robotino.
4. **Análisis de Resultados**.- Someter el funcionamiento del sistema a diferentes pruebas de funcionamiento verificando su calidad y utilidad. Corregir errores en los sistemas mecánicos o electrónicos y de programación.
5. **Conclusiones**.- Conclusiones encontradas durante el desarrollo del proyecto.
6. **Anexos**.- Son códigos fuentes, capturas de pantalla, hojas técnicas, planos e información importante adicional al desarrollo del proyecto.

Los seis puntos descritos constituyen el proceso del proyecto y cada tema será desarrollado más adelante.

CAPÍTULO 1
HIPÓTESIS Y TESIS

La hipótesis busca plantear un resultado desconocido para el tema de investigación, donde la teoría se plantea de forma práctica, la hipótesis puede ser aceptada o rechazada al encontrar los resultados del trabajo de investigación, es decir la hipótesis es una investigación con resultados desconocidos.

La tesis busca plantear un resultado conocido y comprobado para el tema de investigación, plantear la tesis implica saber a dónde se dirige y que resultados se obtendrá al final de la investigación.

HIPÓTESIS

- ¿Es posible diseñar e implementar un sistema manipulador de objetos, modular al Sistema Robot Móvil Robotino de FESTO, por medio de un sistema de control y adquisición de datos y utilizando las herramientas con las que cuenta el Robotino?

TESIS

- Aplicando conocimientos de Mecánica, Electrónica e Informática, se busca desarrollar un Sistema Robot de Transporte Guiado Autónomo, AGV (Automated Guided Vehicle), dotando al Robotino con la capacidad de transportar objetos, y de incrementar un grado de libertad en sus movimientos.

1.1. Justificación

Actualmente los Laboratorios de Electrónica, cuentan entre sus implementos de robótica móvil con los Robotinos marca Festo. Un Robotino Festo en su diseño original posee tres grados de libertad, en desplazamiento dos grados de libertad y el giro sobre su propio eje un grado de libertad. El presente proyecto tiene por objetivo agregar un grado de libertad al Robotino y la opción de manipular objetos con lo cual se brindará una mayor funcionalidad al Robotino.

Las líneas de investigación y aplicación del proyecto son: Robótica, Mecánica y Sistemas de Control. Se busca de esta manera la elaboración de un par cinemático prismático modular al Robotino Festo, que tendrá la capacidad de manipular y trasladar objetos. Con la elaboración del manipulador de objetos se busca dotar de un implemento más para el laboratorio de robótica móvil, sin inferir en mayores gastos y a través de la investigación.

1.2. Objetivos

1.2.1. Objetivo General

- ✓ Expandir la funcionalidad del robot móvil Robotino de Festo adicionando un grado de libertad y un módulo para la manipulación de objetos

1.2.2. Objetivos Específicos

- Construir un módulo mecánico con la capacidad de ser adaptado al chasis y que su control sea transparente en el software de programación del Robotino de Festo.
- Diseñar un par cinemático prismático de un grado de libertad.
- Diseñar el sistema de control electrónico para la pinza manipuladora y el par cinemático prismático.
- Controlar el mecanismo de elevador de forma continua, por medio de un motor a pasos para obtener más de 2 posiciones.
- Implantar el módulo, dando al Robotino un grado de libertad extra y la capacidad de manipular objetos.
- Realizar pruebas del módulo.

1.3. Alcances

Diseño y desarrollo del módulo manipulador de objetos, con la capacidad de:

- Transferencia de movimiento circular a movimiento lineal en forma vertical, por medio de un tornillo sin fin.
- Adaptar una pinza con la capacidad de capturar objetos con un ancho no mayor a 5 cm con un peso de 20 gramos.

- Diseño y desarrollo de un sistema electrónico de control y adquisición de datos, basado en un microcontrolador: Adquirir señales de 24 V_{DC}, Desarrollo del programa para el microcontrolador, Controlar un motor de pasos y un servomotor.
- Diseño y desarrollo de un bloque de funciones para controlar el moduló por medio del programa Robotino View.
- Adaptabilidad entre el módulo manipulador de objetos y el Robotino de FESTO.

CAPÍTULO 2
ESTADO DEL ARTE

El presente capitulo es un estudio de la situación contemporánea del desarrollo de la robótica, mecánica y sistemas de control, con el objetivo de documentar información preliminar para la elaboración del proyecto de tesis planteado, de esta manera buscar una solución con herramientas acordes a la ingeniería electrónica y aplicarlas en el desarrollo del proyecto.

2.1. Robótica

La robótica es una ciencia que busca dar soluciones inteligentes y autónomas a tareas que naturalmente son realizadas por el hombre. Los Robots son el resultado de una construcción de conocimiento y experiencias de muchos años de evolución de ideas, conceptos y tecnología.

La historia, definición y tipos de Robot se estudiara brevemente, luego el estudio se enfoca a la aplicación de un determinado tipo de robot, el Robot Manipulador Industrial, su definición, morfología, cinemática, grados de libertad y criterios de un manipulador industrial.

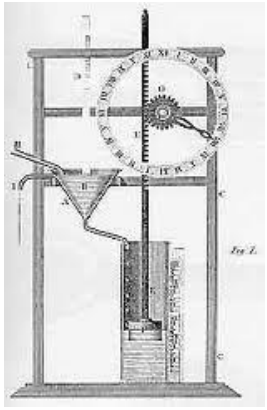
2.1.1. Historia

La historia de la robótica empieza con la búsqueda del hombre por crear mecanismos con movimientos autónomos, motivados por la curiosidad, por demostrar ingenio o capricho de algún rey o emperador pero las primeras máquinas con movimientos autónomos se registran incluso desde antes de Cristo. Siendo los primeros diseños puramente mecánicos y no existen mayores detalles de su funcionamiento pero han sido reconocidas y recordadas por el impacto que causaron.

Algunas de las más famosas creaciones se citan en el libro (BARRIENTOS, PEÑIN, & BALAGUER, 2007, pp. 3,4). Que a continuación se resume en los hechos más relevantes:

- 270 a.C. Autor Ctesibius: Clépsidra y órgano de agua. Aplicaciones de la neumática e hidráulica para la producción de los primeros relojes y órganos de agua.

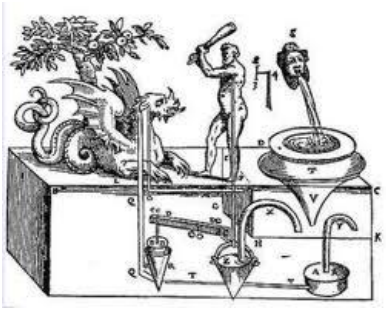
Fig. 1. Reloj de Agua



Fuente: (From Cave Paintings to the Internet, 2006)

- 62 a.C.- Autor Heron de Alejandría. Teatro automático, en el teatro automático las figuras cambiaban de posición, los pájaros cantaban, se oían las trompetas y los animales bebían del agua, entre otros movimientos.

Fig. 2. Teatro Automático



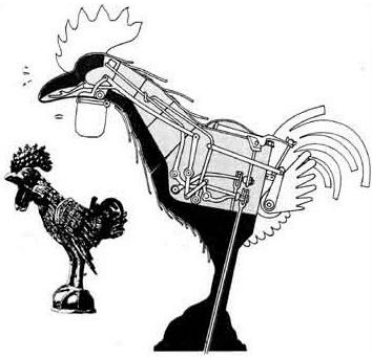
Fuente: (Robots de Inspiración Biológica, 2005)

- 87 a.C.- Autor Escuela de Posidonio. Mecanismo de Antikythera, mecanismo para el cálculo de la posición de los astros. Resulta ser el ejemplo más antiguo de procesamiento de información a través de engranajes.

En la época A.C. el ingenio para construcción y desarrollo mecánico permitió conocer máquinas y herramientas para el entretenimiento y la investigación de astros. Inspirados en la curiosidad del hombre siguen evolucionando junto con la técnica para desarrollar nuevos mecanismos entre los que se destacan:

- 1352.- Autor Desconocido. Gallo de la Catedral de Estrasburgo, es el reloj medieval más famoso y elaborado. Estuvo operativo hasta 1789. Aparecía en compañía de otras doce figuras representando a los apóstoles, movía las alas, levantaba la cabeza y cacareaba tres veces.

Fig. 3. Gallo de la Catedral de Estrasburgo



Fuente: (Autómatas en la Historia, 2004)

- 1500.- Autor Leonardo Da Vinci. León Mecánico, Construido en honor del rey de Francia, Luis XII alrededor del 1500, aunque no se conservan planos del mismo. Se cree que ante el rey dio unos pasos, levanto la garra y se abrió el pecho para enseñar el escudo de armas del rey.
- 1900.- Autor Leonardo Torres Quevedo. Diseña un conjunto de mecanismos capaces de resolver ecuaciones polinómicas por procedimientos estrictamente mecánicos.
- 1912.- Máquina de jugar al ajedrez, fue un sistema mecánico capaz de jugar partidas de Torre y Rey contra Rey, para este tipo de partidas, se pueden dar una serie de reglas que se aseguran jaque mate en un número determinado de movimientos.

Un momento histórico que no se puede olvidar es 1923, la palabra ROBOT es popularizada por Karel Capek en su obra R.U.R. (Rossum´s Universal Robots). Que hasta entonces no era más que una obra de ciencia ficción.

Hasta este punto la robótica o el origen de lo que sería conocido como la Robótica, estaba orientada hacia una ciencia puramente mecánica. A partir de 1947 con la invención del transistor la electrónica tiene un gran desarrollo y una de las aplicaciones son los sistemas de control electrónicos en la robótica.

El desarrollo tecnológico de la robótica nos presenta una evolución acelerada de la robótica que revisaremos brevemente a continuación.

- En 1954, el inventor británico C.W. Kenward solicita la primera patente de un dispositivo robótico, dicha patente fue emitida en el Reino Unido en 1957.
- En 1972, Nissan forma la primera asociación de robótica en el mundo. En 1974 se forma el Instituto de Robótica de América (RIA).
- En 1973, la compañía Sueca ASEA construyó el primer robot de accionamiento totalmente eléctrico llamado Robot IRb6, al año siguiente en 1974 construye el Robot IRb60.
- En 1981, se desarrolla en la Universidad Carnegie Mellon un diseño de robots de accionamiento directo, los motores se acoplan directamente a las articulaciones sin necesidad de reductores, lo que permite movimientos más rápidos y precisos.
- En 1982, el profesor Makino de la Universidad Yamanashi de Japón, desarrolla el concepto de robot SCARA (Selective Compliance Assembly Robot Arm).
- En 1996, la empresa Honda presenta el P-2 y P-3 robots bípedos con la apariencia de un traje espacial.
- En 1999, la empresa Sony desarrolla el primer robot mascota, un perro con la capacidad de ser entrenado, comportamiento autónomo, responde a estímulos externos de su dueño.
- En 2002, honda desarrolla una versión mejorada del P-2 y P-3 llamada ASIMO.

Actualmente ASIMO (acrónimo de "Advanced Step in Innovative Mobility"- paso avanzado en movilidad innovadora), es un robot humanoide que fue desarrollado por la empresa HONDA, posee la capacidad de caminar desde 1,6 km/h a 2,5 km/h y correr a 3km/h. No se puede olvidar a CURIOSITY, un robot desarrollado por la NASA y que se encuentra recorriendo Marte en la actualidad.

La robótica y sus aplicaciones se encuentran en desarrollo constante, buscando brindar soluciones en procesos y tareas en diferentes aéreas como: Industria, Seguridad, Medicina, Ensamblaje etc. Un sistema autónomo o robot cumple con la característica de realizar tareas repetitivas y programadas, por ejemplo: Manipulación o Trasporte.

2.1.2. Tipos de Robots

Un Robot de acuerdo a su característica principal y utilidad recibir un nombre, algunos tipos de robots se mencionan en (BARRIENTOS, PEÑIN, & BALAGUER, 2007, p. 17) que a continuación se mencionan.

- Robot aéreo.
- Robot asistencial, inspección y explorador.
- Robot humanoide.
- Robot manipulador estático y móvil.
- Robot Marino y Submarino.
- Robot Teledirigido.
- Mini y micro Robot.
- Entre muchos más…

De esta lista el sistema o robot que nos interesa conocer es el manipulador móvil mejor conocidos como Sistema de Transporte Guiado Automático AVG (Automated Guided Vehicle). Que es el término utilizado por (FESTO Didactic GmbH & Co. KG, 2008).

2.1.3. Definición de Robot

El robot en aplicaciones industriales posee una definición de acuerdo a la solución que brinda dentro de un proceso. Para encontrar una definición se tendrá presente algunas citas textuales.

Según (Etimologías, 2006) , la definición etimológica del término Robot se puede expresar como: "La palabra Robot viene del checo robota (trabajo forzado) y rabota (servidumbre). La popularizó el dramaturgo checo Karel Capek por su obra "Los Robots Universales de Rossum" (Rossum´s Universal Robots) publicada en 1920."

La (Real Academia de la Lengua Española, 2001) propone una definición formal para la palabra robot esta definición dice así: "Maquina o ingenio electrónico programable, capaz de manipular objetivos y realizar operaciones antes reservadas sólo a las personas."

Y finalmente según (Wikipedia, La Enciclopedia Libre, 2006): "Un robot es una entidad virtual o mecánica artificial. La robótica concentra 6 áreas de estudio: La mecánica, el control automático, la electrónica, la informática, la física y la matemática como ciencias básicas."

Entonces se puede decir que un robot es una máquina que está conformada por un sistema mecánico y electrónico. Para su desarrollo y diseño se utiliza herramientas como el control automático, la informática, la física y matemática. Un robot cumple con tareas programadas de acuerdo a las aplicaciones del robot, por ejemplo: Seguir una trayectoria, Manipular objetos, reconocimiento de patrones, etc. De acuerdo a su característica principal el robot recibe una definición por ejemplo: Robot Seguidor de Línea, Robot Manipulador de objetos, etc.

2.1.4. Definición de Robot Industrial Manipulador

El robot industrial manipulador es un tipo específico de robot que tiene por objetivo, capturar y manipular objetos, piezas, tornillos, etc. Con la capacidad de trasladar los objetos de un lugar a otro. Según la norma ISO (Organización Internacional de Normalización) y RIA (Robotic Industries Association), plantean las siguientes definiciones al termino Robot Industrial Manipulador.

Para la organización RIA (Robotic Industries Association): "Un robot industrial es un manipulador multifuncional reprogramable, capaz de mover materias, piezas,

herramientas o dispositivos especiales, según trayectorias variables, programadas para realizar tareas diversas.”

Y según la Norma (ISO_8373, 1998): “Un robot manipulador industrial (ISO): manipulador de 3 o más ejes, con control automático, reprogramable, multiaplicación, móvil o no, destinado a ser utilizado en aplicaciones de automatización industrial. Incluye al manipulador (sistema mecánico y accionadores) y al sistema de control (Software y hardware de control y potencia)”

La misma norma ISO define ciertas condiciones, que se resumen en (BARRIENTOS, PEÑIN, & BALAGUER, 2007) a continuación:

- Reprogramable: Aquellos en los que los movimientos programados o las funciones auxiliares pueden cambiarse sin modificación física.
- Modificación Física: Modificaciones de la estructura mecánica o del sistema de control (se excluyen cambios en los soportes de memoria: disco, cinta, rom, etc…)

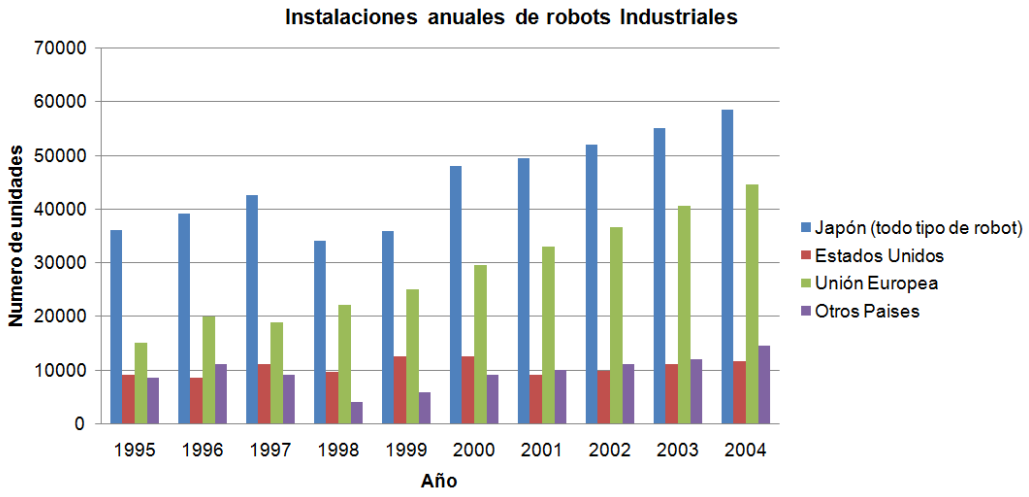
Según la Empresa FESTO al Robot Manipulador se le denomina como AVG según sus siglas en ingles (Automated Vehicle Guided) o Sistema de Transporte Guiado Automático.

2.1.5. Situación Actual

En la época contemporánea la robótica se vuelve una industria, la investigación desarrolla aceleradamente la creación de nuevas aplicaciones y nuevos tipos de robots. Se puede decir que por cada necesidad en un proceso industrial se desarrolla un nuevo tipo de robot.

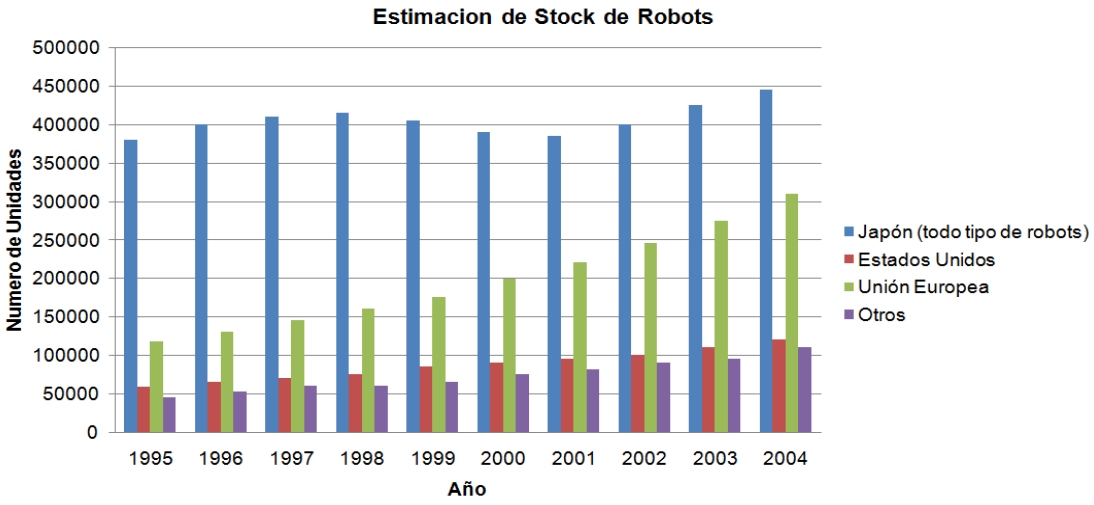
Algunas estadísticas de este crecimiento se revisarán a continuación.

Fig. 4.- Instalaciones anuales de robots industriales



Fuente: (TORRES POMARES & PUENTE, 2005, pág. 15).;
Elaborador por: Cristian Cuji.

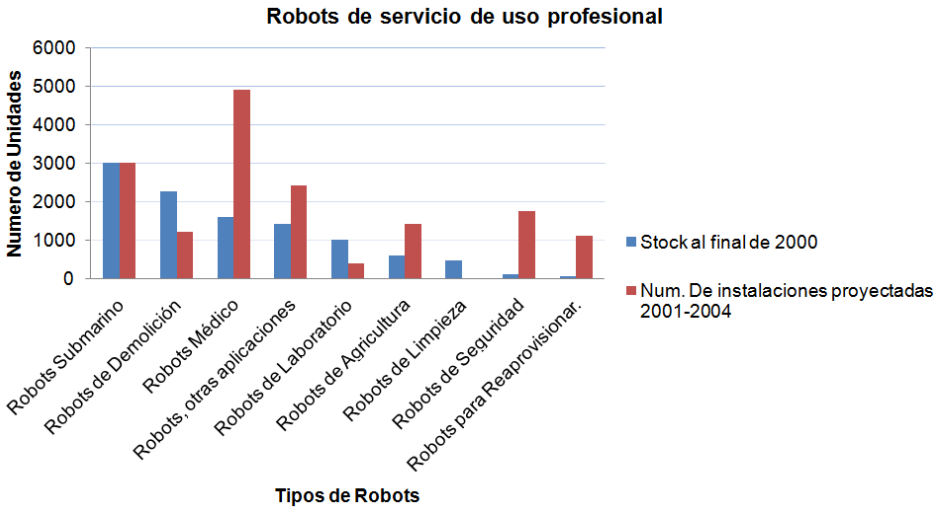
Fig. 5.- Estimado de stock de robots



Fuente: (TORRES POMARES & PUENTE, 2005, pág. 16)

Elaborador por: Cristian Cuji.

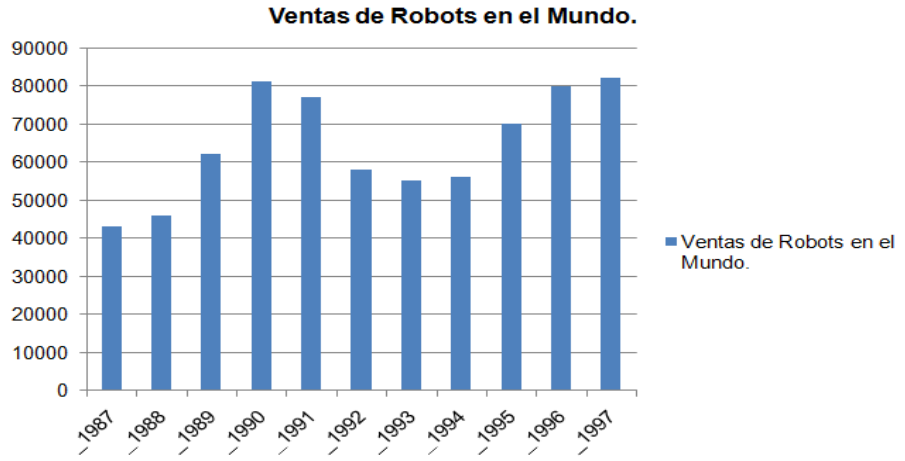
Fig. 6.- Robots de servicio de uso profesional



Fuente: (TORRES POMARES & PUENTE, 2005, pág. 17)

Elaborador por: Cristian Cuji.

Fig. 7.- Ventas de Robots en el Mundo



Fuente: (Present and prospects of Robotics, 2002)

Elaborador por: Cristian Cuji.

Como se aprecia en la Fig. 7.- Ventas de Robots en el Mundo En 10 años de producción desde 1987 a 1997, han existido incrementos y descensos en las ventas, pero en la actualidad se busca dar cada vez más aplicaciones a los robot en áreas de manufactura, ensamblaje, otras áreas industriales y de producción.

El siguiente texto es un extracto del Articulo "Un robot me ganó el trabajo" publicado por CNN "Cable News Network (Cadena de Noticias por Cable)" en 2011 septiembre 17 atreves de (CNN expansión, 2011). El artículo hace referencia con datos actuales al número de robots existentes en labores antes desempeñadas por humanos y también al número proyectado de unidades de robots hasta el 2025.

"Un robot me ganó el trabajo"

La Federación Internacional de Robótica (IFR, por sus siglas en inglés), en su estudio Global World of Robotics 2011**,** la IFR sostiene que luego de un declive masivo durante 2009 la industria de la robótica industrial y de servicios mantendrá un crecimiento sostenido, al menos durante los siguientes cinco años. Tan solo en 2010, la **IFR (International Federation of Robotics)**, estima que se vendieron más de 118,337 robots alrededor del planeta**.** La industria automotriz y de electrónica de consumo se coloca como los principales compradores de estas máquinas, al adquirir 52% de las unidades.

Y si bien Estados Unidos, Japón y Alemania se mantuvieron como los tres países con el mayor consumo de robots industriales durante 2010, la IFR advierte sobre un crecimiento e inversión exponencial en naciones como China, Corea, India y Brasil. Quizá la cifra de 120,000 robots vendidos en 2010 puede no significar mucho. Pero para la IFR el mercado de la robótica apenas está por "explotar". El organismo asegura que el mercado tiene un valor por encima de los 17,500 millones de dólares y que actualmente existe más de un millón 30,000 robots industriales trabajando alrededor del planeta. Y el número únicamente seguirá elevándose. Para 2011 la IFR estima que se venderán más de 139,000 unidades y proyecta que para 2014 la base instalada de robots industriales oscile entre 1,300,000 y un 1,600,000**.**

El reporte Global Trends 2025: A Transformed World, de Consejo de Inteligencia Nacional de Estados Unidos (NIC, en inglés) sentencia que para 2025 los robots habrán sustituido a los seres humanos no sólo como empleados en las líneas de manufactura, sino incluso como trabajadores en diversas áreas de servicio y asistencia. (CNN expansión, 2011)

2.1.6. Robótica Móvil

Los Robots Móviles pueden desplazarse cumpliendo múltiples tareas adicionales en lugares remotos, dichos sistemas móviles han obtenido grandes avances en el mercado, el desplazamiento es la principal ventaja sobre sus predecesores los robots estacionarios. Este tipo de sistemas hoy en día, están siendo aplicados en diversas áreas, tareas y servicios, etc. Donde se requiera de acciones cíclicas o mecánicas un robot es una solución eficiente. La idea de utilizar un robot es solucionar tareas que el hombre no desea hacer, por riesgosa, repetitiva o por evitar errores provocados por cansancio.

Fig. 8. Robots Móviles de Festo y Pathfinder Rover – Sojourner

Fuente: (FESTO B. W., 2007)
Elaborador por: Cristian Cuji.

2.1.7. Robotino de Festo®

2.1.7.1. Festo

Es una empresa Alemana, líder en sistemas de automatización tanto para el área industrial como didáctico, es también fabricante de módulos de enseñanza en: Neumática, Hidráulica, Sensores y Actuadores, PLC, Automatización, Mecatrónica, Robótica, Control de procesos y Sistemas industriales de producción para simulación de proceso.

Fig. 9. Logo FESTO. CIA. Ltda. Alemania

Fuente: (FESTO A. , 2012)
Elaborador por: Cristian Cuji.

El desarrollo de robots móviles para investigación y para propósitos específicos ha llevado a que la compañía Festo y Festo Didactic, desarrollen un robot exclusivo para investigación y el entrenamiento de estudiantes, por estos motivos es desarrollado ROBOTINO que posee su propio lenguaje de programación que es el Robotino View y cuya versión actual es Robotino View 2.8.4.

2.1.7.2. Robotino

Es un robot móvil disponible en el mercado de Festo Didactic. Se utiliza tanto para la educación y la investigación, incluyendo las competiciones como RoboCup. Cuenta con una base omnidireccional, sensores de choque, sensores infrarrojos, sensores de distancia y una cámara en color VGA. El diseño de Robotino es modular, y puede ser fácilmente equipado con una variedad de accesorios, gracias a un módulo de entradas y salidas.

Fig. 10. Robotino de Festo

Fuente: (FESTO B. W., 2007)

2.2. Mecánica

2.2.1. ¿Qué es mecánica?

Según el Diccionario de la (Real Academia de la Lengua Española, 2001) la palabra Mecánica se puede definir como:

“f. Parte de la física que trata del equilibrio y del movimiento de los cuerpos sometidos a cualquier fuerza.”

“f. Aparato o resorte interior que da movimiento a un ingenio o artefacto.”

La Mecánica se define como la ciencia que describe y predice las condiciones de reposo o movimiento de los cuerpos sometidos a la acción de fuerzas. Se divide en tres partes:

- Mecánica de cuerpos rígidos.
- Mecánica de cuerpos deformables.
- Mecánica de fluidos.

La Mecánica de cuerpos rígidos se subdivide en estática y dinámica. La estática estudia los cuerpos en reposo y la dinámica los cuerpos en movimiento. Para el estudio de la mecánica de cuerpos rígidos se supone que los cuerpos son perfectamente rígidos o sólidos, es decir no sufren deformación, pero la estructuras y maquinas reales nunca lo son y generalmente se deforman cuando son sometidos bajo fuerzas o cargas. Estas deformaciones casi siempre pequeñas no afectan las condiciones de equilibrio o de movimiento de la estructura, pero son importantes cuando se tiene en cuenta la resistencia de la estructura a fallas.

2.2.2. Principios fundamentales de la mecánica

La mecánica está basada sobre seis principios fundamentales, (BEER P, JOHNSTON E, & EISENBERG R, 2005), realiza un estudio de los principios que servirán como base para el presente estudio:

Ley del Paralelogramo: Dos fuerzas aplicadas a un cuerpo pueden ser remplazadas por una fuerza resultante, se obtiene al trazar una diagonal del paralelogramo que tiene los lados iguales a las fuerzas dadas.

El principio de transmisibilidad. Una fuerza aplicada a un cuerpo rígido es equivalente a la fuerza generada en otro punto con la misma magnitud y la misma dirección, es decir que tengan la misma línea de acción.

Las tres leyes fundamentales de Newton

Primera Ley: Si la fuerza resultante que actúa sobre una partícula es cero, la partícula permanecerá en reposo (Si originalmente estaba en reposo) o se moverá con velocidad constante en línea recta (si originalmente estaba en movimiento).

Segunda Ley: Si la fuerza resultante que actúa sobre una partícula no es cero, la partícula tendrá una aceleración proporcional a la magnitud de la resultante y en la dirección de esta.

$$F = ma.$$

F (fuerza resultante sobre la partícula), m (Masa), a (Aceleración).

Tercera Ley: Las fuerzas de acción y reacción de cuerpos en contacto tienen la misma magnitud, la misma línea de acción y sentidos opuestos.

La Ley de gravitación de Newton. Establece que dos partículas de masa M y m se atraen mutuamente con fuerzas iguales y opuestas F y -F.

2.2.3. Utilidad de la mecánica en el proyecto

Los conocimientos básicos de Mecánica son necesarios, porque el proyecto plantea la elaboración de un artefacto o aparato con la capacidad de capturar y desplazar objetos. Durante el diseño y el desarrollo del artefacto se debe tomar en cuenta: Materiales de Construcción, Sistema de Transmisión de Movimiento Mecánico y Herramientas de Diseño Asistido por Computadora.

Los metales son la materia prima en la elaboración de herramientas para diferentes industrias (construcción, automovilística, metalmecánica, etc.). Desde los primeros sistemas autónomos, hasta los sofisticados sistemas actuales, el uso de metales y su aplicación a los sistemas mecánicos son un pilar fundamental en el diseño y construcción del sistema. El uso de los metales como herramientas para el diario vivir del hombre es históricamente conocido. En Egipto y las ruinas de Nínive se han

encontrado algunas herramientas que eran utilizados para tallar el granito y pórfido, es decir estas herramientas tendrían una antigüedad de 6000 años.

Fig. 11. Evolución del Uso de Materiales Metálicos

Fuente: (Zenteno, Ignacia Carvallo, 2010)

Hoy en día las aleaciones de metal son utilizadas con fines de obtener diferentes características por ejemplo de flexibilidad, rigidez, durabilidad, etc. Como por ejemplo: Las Microláminas de Carbono, ligeros como el papel pero 30 veces más fuerte que el acero.

2.2.4. Materiales de Construcción para Sistemas Mecánicos

De acuerdo con (ORÚS ASSO, MATERIALES DE CONSTRUCCIÓN, 1977). Los metales más utilizados en el mercado nacional son Cinc, Cobre, Estaño, Aluminio, Hierro. El objetivo es conocer algunas características de los materiales de la construcción de sistemas mecánicos tradicionalmente utilizados.

Fig. 12. Metales

Fuente: (ZAMBRANO Morales, Yenny Zoraida, 2010)

Algunas características de los metales son y sus definiciones según él (DICCIONARIO DE LA LENGUA ESPAÑOLA 10maE, 2012) se consideran a continuación:

- Densidad.- Magnitud que expresa la relación entre la masa y el volumen de un cuerpo. Su unidad en el Sistema Internacional es el kilogramo por metro cúbico (kg/m^3).
- Dureza.- Resistencia que opone un mineral a ser rayado por otro.
- Elasticidad.- Propiedad general de los cuerpos sólidos, en virtud de la cual recobran más o menos completamente su extensión y forma, tan pronto como cesa la acción de la fuerza que las deformaban.
- Tenacidad.- Que opone mucha resistencia a romperse o deformarse.
- Conductividad.- Propiedad que tienen los cuerpos de transmitir el calor o la electricidad.

2.2.5. Propiedades de los metales

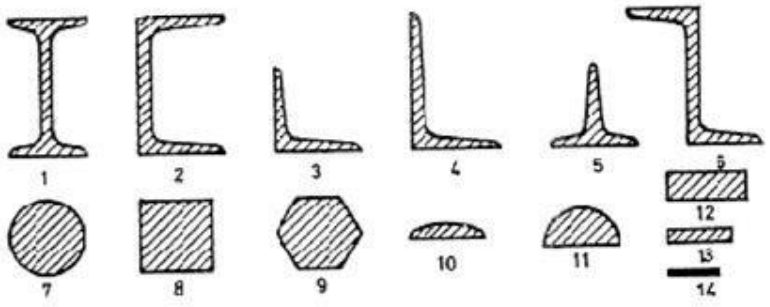
Según (ORÚS ASSO, MATERIALES DE CONSTRUCCIÓN, 1977) pág. 365 y 366: "La forma se la comunica a los metales fundiéndolos colando en moldes donde se solidifica y enfría según su mayor o menor fusibilidad; por medios mecánicos a elevada temperatura, según su forjabilidad; en frio, según su maleabilidad, y finalmente por separación y acoplamiento, según las propiedades de fácil corte y soldadura."

a) Fusibilidad.- Depende de su punto de fusión, cuando el metal llega a una determinada temperatura este se funde, es decir se vuelve liquido. En este estado es donde se puede realizar las aleaciones.
b) Forjabilidad.- Con calentamiento es la capacidad de soporte del material en estado sólido sin cambiar su forma. Por medio de calor generar una variación de su forma por medio de acciones mecánicas de martillos, laminadores y prensas, sin pérdida de cohesión.
c) Maleabilidad.- A temperatura ambiente, es la propiedad de los metales de poder modificar su forma por acciones mecánicas de martillo, estirado y laminado. La mayor diferencia entre forjabilidad y maleabilidad es que la forjabilidad se puede dar forma hasta donde se desee sin perder propiedades, mientras que con

maleabilidad se limita dar forma, hasta un punto donde el metal se vuelve quebradizo.

d) Ductilidad.- Es la capacidad alargar un cuerpo en la dirección de su longitud, depende de la tenacidad, es necesario que un material metálico tenga un límite bajo de elasticidad, sean resistentes y medianamente blandos.
e) Tenacidad.- Es la resistencia a la rotura, por medio de la atracción que tienen los cuerpos debido a la cohesión de las moléculas que los integran, expresándose en kilogramos por milímetro cuadrado.
f) Facilidad de corte.- Es la propiedad de ser separado en pedazos por medio de herramientas cortantes. Un material al no poseer esta propiedad o ser muy duros se rompe en pedazos irregulares, y al poseer características muy blandas se vuelve como pasta y se adhiere a la herramienta de corte, Ejemplo(el Plomo)
g) Soldabilidad.- Es la propiedad de poder unir por presión dos metales, hasta formar un trozo único, esta unión puede hacer solo a elevada temperatura.
h) Oxidabilidad.- Es la capacidad de sufrir corrosión por medio del oxigeno, todos los metales se oxidan menos los más nobles (oro, plata y platino.)

Fig. 13. Figuras comerciales del hierro



Fuente: (Tecnología Mecánica, 2012)

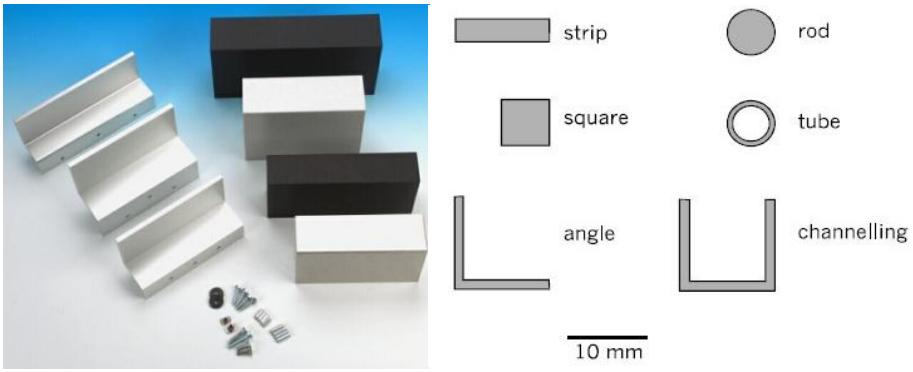
2.2.6. Descripción del Aluminio

Aluminio pertenece al grupo 13 de la tabla periódica junto con el Boro, Su símbolo es Al y es un elemento químico que se lo encuentra en forma metálica, su número atómico 13. Es muy abundante en el medio ambiente, por esta razón es muy común verlo en diferentes presentaciones. Desde tubos de aluminio hasta hojas de aluminio. En la

naturaleza se encuentra combinado, formando arcillas; en forma de óxidos hidratado, $Al_2O_3.H_2O$, la bauxita y la criolita o fluoruro de aluminio y sodio P_6AlNa_3.

Por lo general el aluminio se vende en longitudes de dos metros y existen en una variedad de tamaños y secciones transversales. El siguiente dibujo muestra algunas de ellas.

Fig. 14. Secciones trasversales del stock de aluminio



Fuente: (BISHOP, 2007)

Propiedades

Es un Metal Blando brillante con matiz ligeramente azulado de estructura fibrosa:

- Blando en 2.9 de la escala de Mohs.
- Muy ligero, densidad 2.7 veces mayor que la del agua.
- Punto de fusión 658°C
- Posee una buena conductividad eléctrica, que se encuentra entre los 34 y 38 m/Ω mm^2, así como también tiene una gran conductividad térmica, entre los 80 a 230 W/ m.K.
- Es muy dúctil y maleable, pudiendo obtenerse hilos y hojas.
- La resistencia mecánica depende del grado de pureza.
- Es resistente a la corrosión, gracias a la capa protectora característica de óxido de aluminio, resiste a los productos químicos, puede estar expuesto a la intemperie, al mar, etc.
- Es el tercer elemento en cuanto a abundancia en la corteza terrestre, por detrás del oxígeno y el silicio.

El aluminio se protege con una capa de óxido invisible que se aplica como un barniz por medio de electrolisis y se puede colocar obteniéndose el aluminio anodizado, por esta razón el aluminio es resistente a la oxidación y es muy empleado en decoración. Las aguas potables y ácidos le atacan.

Formas comerciales

Angulares:

- Lados iguales desde 10 x 10 x 2 hasta 60 x 60 x 6
- Lados desiguales 10 x 20 x 2 hasta 25 x 50 x 5

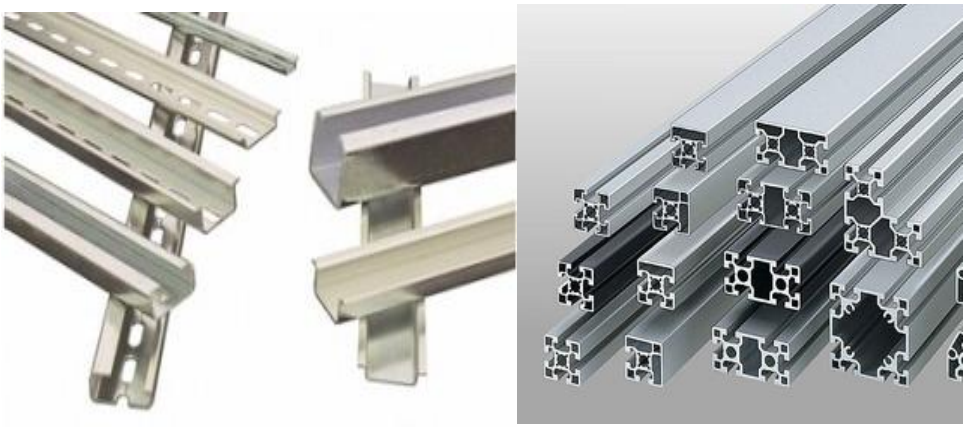
Chapas:

- Desde 0.2 mm hasta 5 mm aumentando de 0.5mm

Alambres:

- Desde 1 hasta 50 mm de diámetro.

Fig. 15. Presentaciones comerciales del Aluminio



Fuente. (Direct Industry, 2012)

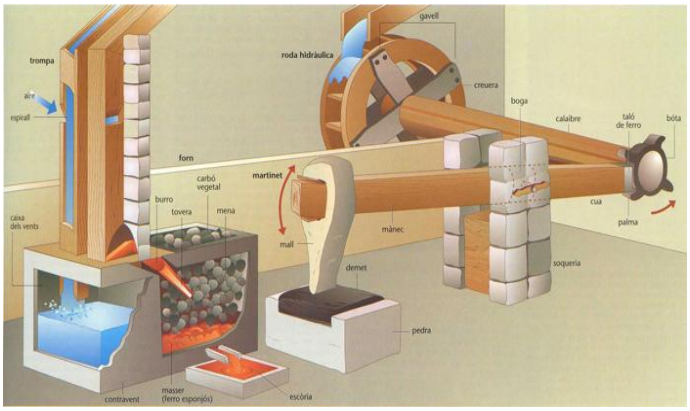
2.2.7. Descripción del Hierro Dulce

El hierro se encuentra presente naturalmente en estado sólido, es obtenido por el procedimiento conocido como forjas catalanas. Se lo puede observar en estado líquido solo en hornos sometiendo al hierro a elevadas temperaturas.

Forjas Catalanas

Este es un proceso por el cual se obtiene hierro dulce o forjable y también el acero, pero es solo aplicable a minerales muy ricos. Se introdujo en la Edad Media en el siglo XI y consistía básicamente en un horno que quemaba carbón mezclado con aire a presión con el fin de alcanzar una temperatura muy alta, unos 1000º C, con la que se derretía el mineral y se lograba la separación del hierro (la mena) de las impurezas (ganga).

Fig. 16. Forjas Catalanas



Fuente: (Grup Enciclopedia Catalana, 2010)

El objetivo de este procedimiento es conseguir hierro dulce que es el utilizado en piezas de hierro solidas, utilizadas generalmente en piñones, engranes, tornillos sin fin, etc.

2.2.8. Aplicaciones del hierro dulce

Se necesita una gran cantidad de calor para separar el hierro de las impurezas, este es un procedimiento que hoy sigue siendo utilizado por industrias para obtener diferentes tipos de herramientas y artículos como por ejemplo remaches, piñones, pernos, engranes, etc. De diferentes dimensiones y características. Una de las características por las cuales se ha utilizado metales en el desarrollo de sistemas mecánicos es rigidez, firmeza, durabilidad, elasticidad, dilatación. En todo sistema mecánico uno de los factores que se presenta es la fuerza o presión a la que va ser sometido.

Fig. 17. Aplicaciones del Hierro en la Actualidad

Fuente: (Metal Mecanica, 2011)

2.2.9. Aplicación de Sistemas Mecánicos

Un Sistema Mecánico es un conjunto de elementos generalmente constituido por piezas solidas, tiene por objetivo transformar o transmitir el movimiento desde las fuentes que lo generan a distintos tipos de energía como por ejemplo movimiento, pueden estar asociados a sistemas eléctricos. El diseño del mecanismo exige escoger los elementos, materiales y medidas de cada elemento que conforma el sistema mecánico del proyecto.

2.2.9.1. Sistemas de Transmisión de movimiento

Según (ROLDÁN VILORIA, 2002) los sistemas de transmisión de movimiento poseen dos elementos:

- Elemento de entrada del mecanismo (elemento motriz)
- Elemento de salida (elemento conducido).

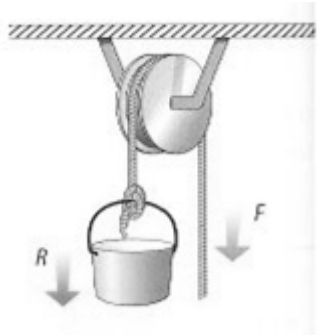
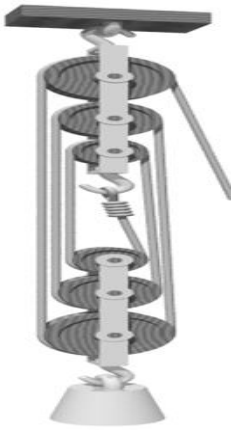
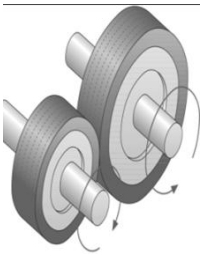
El elemento de entrada debe coincidir con el tipo de movimiento que tiene el elemento de salida. Además los mecanismos de transmisión pueden ser agrupados en dos grandes grupos:

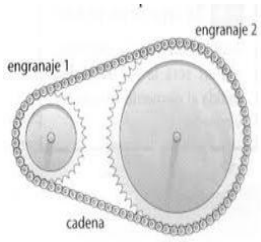
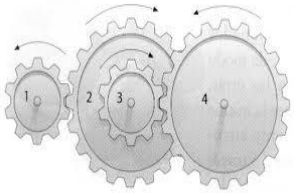
1. Mecanismos de transmisión circular.- En este caso, el elemento de entrada y el elemento de salida tienen movimiento circular. Ejemplo: Los sistemas de engranajes.

2. Mecanismos de transmisión lineal.- En este caso, el elemento de entrada y el elemento de salida tienen movimiento lineal. Ejemplo: La palanca.

El movimiento del eje reductor se transmite a la maquina a través de:

Tabla- 1. Sistemas de Transmisión de Movimiento Mecánico

N°	Sistema	Descripción	Observaciones
1	Sistema de Polea Fija 	La polea fija es un artefacto que no genera mayor fuerza, solo una mejor posición o más comodidad para ejercer fuerza y levantar cargas.	$FA = FR$
2	Sistema de Poleas Compuestas 	Es un sistema es un conjunto de poleas fijas y móviles que si genera una mayor ventaja mecánica para levantar grandes cargas con un bajo esfuerzo. Al sistema de poleas compuestas se denomina Polipasto.	$FA = \frac{FR}{2 * n}$ n: número de poleas fijas del polipasto. FA: Fuerza Aplicada. FR: Fuerza de Resistencia (Peso)
3	Ruedas de Fricción 	Es un sistema donde los ejes de las ruedas se encuentran ubicados en paralelo, las ruedas por efecto de rozamiento transmiten el movimiento de una rueda a la otra, por medio de este mecanismo se puede modificar características de velocidad y sentido de giro.	$n1 * d1 = n2 * d2$ n_1 = velocidad de la rueda motriz. n_2 = velocidad de la rueda conducida d_1 = diámetro de la rueda motriz (entrada). d_2 = diámetro de la rueda conducida (salida).

4	Transmisión de engranes con cadena 	Es un sistema de transmisión de movimiento por medio de una cadena sin fin con eslabones que se engranan con ruedas dentadas entre los mecanismos conductor y conducido.	La ventaja principal de la cadena y ruedas dentadas es evitar que es el sistema el deslizamiento del sistema. $FA = FR.$
5	Tren de engranajes 	Es un mecanismo que consiste en la combinación de más de un par de engranajes. Es un sistema de transmisión circular muy común con múltiples y variadas aplicaciones.	El funcionamiento es similar al de ruedas de Fricción, con la ventaja de no sufrir de deslizamientos.
6	Tornillo sinfín y rueda dentada 	Es un mecanismo de transmisión circular compuesto por un tornillo sin fin (elemento motriz) y rosca dentada (elemento conducido). La rosca y el tornillo son perpendiculares entre sí. Este mecanismo no es reversible, es decir, la rueda no puede mover el tornillo porque se bloquea.	La siguiente ecuación es la relación de transmisión: $i = \frac{1}{z}$ Donde Z representa el número de dientes del engranaje.

Fuente: (SEFARAD, 2010) (Aprendamos Tecnología, 2008)
Elaborado por: Cristian Cuji.

2.2.10. Herramientas de diseño asistido por computadora

En el desarrollo de todo proyecto un bosquejo, un plano o mínimas instrucciones expresadas en un dibujo sencillo es necesario para saber qué es lo que se desea realizar. La representación gráfica es el acto de expresar ideas por medio de líneas y marcas impresas sobre una superficie o papel. Un dibujo es una representación de un objeto real, por lo tanto el dibujo es un lenguaje gráfico. Dichas imágenes pueden ser interpretadas por gente de cualquier lugar del mundo, por tal motivo el dibujo recibe el nombre de lenguaje universal.

(JENSEN, SHORT, & HELSEL, 2002) Expresa que un dibujo técnico constituye el punto de partida de todo trabajo profesional, el plano final es el elemento principal en la comunicación entre los involucrados en el diseño y la fabricación del artefacto o sistema. Hace pocos años era común el dibujo técnico manual como herramienta para el diseño de todo tipo de proyectos, saber utilizar las reglas las escuadras y los diferentes tipos de lápices era verdaderamente un arte. La precisión al momento de trabajar dependía del dibujante y un error era fatal ya que podía dañar el esfuerzo de todo un trabajo.

Actualmente las herramientas gráficas asistidas por computadora facilitan el trabajo, es solo necesario tener las ideas básicas del dibujo técnico y por medio de un computador el trabajo del dibujante se facilita, la precisión ya no es problema y los errores pueden ser editados y corregidos rápidamente.

2.2.10.1. Dibujo técnico manual

El dibujo técnico es un método para representar de manera gráfica cualquier tipo de objeto, con el propósito de proporcionar información suficiente para facilitar su análisis, elaborar un diseño y posibilitar la futura construcción. Esta representación se guía por normas fijas y preestablecidas para poder describir de forma exacta y precisa las dimensiones, formas y características del objeto que se quiere construir o reproducir. Tradicionalmente algunas herramientas para realizar dibujos técnicos eras escuadras, regla, compas, tablero, borrador y diferentes tipos de lápices entre muchas otras herramientas.

Fig. 18. Algunas herramientas utilizadas tradicionalmente en el dibujo técnico

Fuente: (La Tienda del Maestro, 2012)

2.2.10.2. AutoCAD

Los computadores personales y el desarrollo de Autodesk AutoCAD substituyo las mesas de dibujo. AutoCAD es una herramienta que consiste en un programa de diseño asistido por computadora para dibujo en dos y tres dimensiones. Actualmente es desarrollado y comercializado por la empresa Autodesk. El término AutoCAD (Computer - Aided Design) o Diseño Asistido por Computadora surge como creación de la compañía Autodesk, teniendo su primera aparición en 1982. Actualmente es un software reconocido a nivel internacional por sus amplias capacidades de elaboración y edición de dibujo digital, planos o la recreación de imágenes en 3D.

Se debe tener en cuenta que un sistema CAD no realiza los gráficos por sí mismo, es necesario que el dibujante indique todos los detalles del diseño al computador. El computador solo cumple las tareas y sirve de asistente para realizar las mismas. El uso de Auto CAD también requiere un grado de capacitación adicional al conocimiento de dibujo técnico, es importante conocer o capacitarse con el uso del Software Auto CAD en la versión que se desee. Según (GUTIÉRREZ SALAZAR & CRUZ LAVERDE, 2008) Auto CAD hoy en día es un estándar de diseño por sus múltiples herramientas entre las que se destacan:

- Agilizar el proceso de creación de bloques, es decir no tener que repetir un dibujo de similares características, un bloque una vez creado está disponible para seguir siendo usado el número de veces que sea necesario en el diseño y por otros proyectos.
- Compartir los diseños entre distintos equipos de diseño que usen versiones anteriores de Auto CAD.
- Generar dibujos altamente complejos, en 3D y poder manipular sus vistas como se desee, por medio de las herramientas de visualización 2D y 3D.
- Utilizar operaciones de conjuntos como: Unión, Sustracción, Intersección e Inferencia en sólidos, regiones o superficies sólidas.
- Revisar, con la integración de cámaras y recorridos, el proyecto como si estuviera ya construido.

Fig. 19. Herramientas y entorno de trabajo Auto CAD

Fuente: AutoCAD

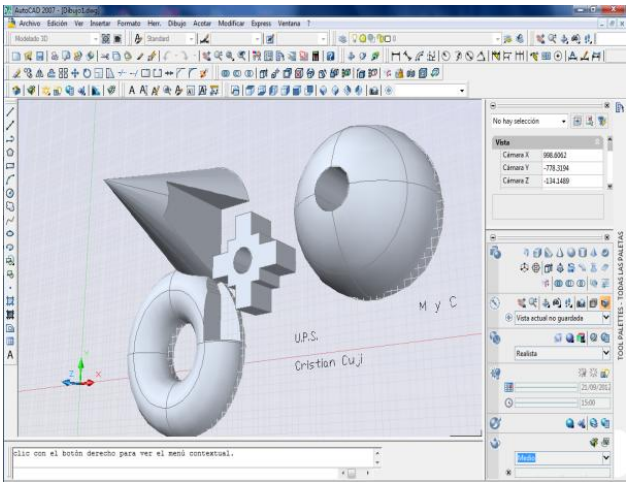
Aplicaciones del Auto CAD

Algunas aplicaciones pueden ser:

- Arquitectura y construcción
- Diseño Eléctrico, Electrónico y Electromecánico.
- Mapeo, Topografía / Carreteras.
- Modelación de terreno.
- Diseño Industrial, Mecánico y de Plomería.

Por ejemplo Auto CAD Electrical es una herramienta exclusiva para el diseño de instalaciones civiles, industriales y planos de acometidas eléctricas. A continuación se puede observar una demostración de estas herramientas.

Fig. 20. Entorno grafico de Auto CAD 3D



Fuente: AutoCAD 2010
Elaborado por: Cristian Cuji

En capítulos posteriores Auto CAD será muy útil para desarrollar el modelo mecánico a implementar para visualizar el diseño en 2D y en 3D. No interesan detalles de funcionamiento, pero se debe considerar los criterios de transmisión de movimiento mecánico, uso de materiales de construcción, ubicación de elementos, etc.

2.3. Sistema de Control

Es un conjunto de elementos electrónicos que permite dar autonomía en sus movimientos al sistema mecánico del manipulador. La parte mecánica de un robot es importante y requiere de gran precisión para obtener movimientos adecuados. Pero los movimientos del robot necesitan de un control electrónico y de una programación previa para que el sistema sea autónomo en su funcionamiento.

Desde la invención del transistor, seguido de los circuitos integrados los sistemas de control electrónicos han sido un gran aporte al desarrollo de la robótica, en la actualidad los dispositivos de control más difundidos son los siguientes: Microcontroladores, PLC (Controlador Lógico Programable), Controladores, Computadores entre otros que cumplen la función de ser el cerebro de un proceso ya permite ejecutar acciones, controlar movimientos, recibir señales, etc. Al implementar un sistema de control se requiere considerar un modelo de arquitectura para el desarrollo del proyecto, actualmente los modelos o arquitecturas más utilizadas son conocidos como sistemas CISC y RISC.

2.3.1. Modelos de Arquitectura CISC y RISC

El uso de dispositivos como teléfonos celulares, computadores, calculadoras y más dispositivos electrónicos ha sido posible gracias al desarrollo de dos arquitecturas que en la actualidad se denominan CISC y RISC. Según Los sistemas CISC son los responsables del desarrollo de toda la familia de procesadores Intel x86. En cambio el sistema RISC es una arquitectura que se presta para realizar investigación, desarrollo de tecnología y presenta características similares pero reducidas en microcontroladores. (MATIC, PIC MICROCONTROLLERS BOOK1, May 15, 2000).

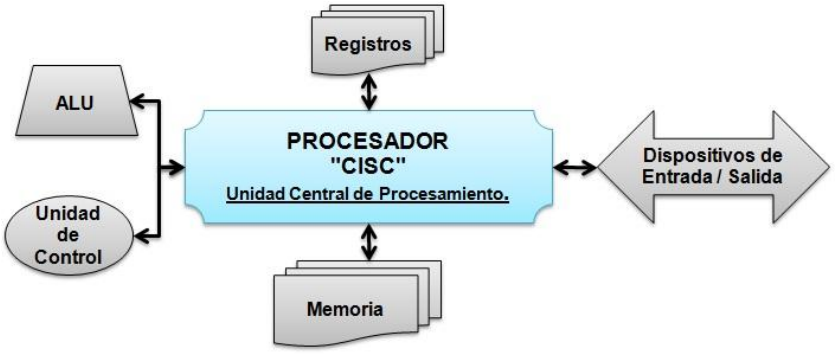
CISC

Complex Instruction Set Computing o Conjunto Complejo de Instrucciones de Computación. Es el modelo de arquitectura más difundido comercialmente, se encuentra en los computadores portátiles y de escritorios tradicionales por ejemplo: Pentium 1, 2, 3, 4, Intel Core Duo, Core 2 Duo, i3, i4, i5, etc. La idea de estos dispositivos es procesar grandes cantidades de información. Estos procesadores poseen las siguientes características:

- Un conjunto de instrucciones muy amplio.
- Permite operaciones complejas entre operando situados en la memoria o en los registros internos.
- Dificulta el paralelismo entre instrucciones.

El termino CISC fue utilizado a raíz del desarrollo de la tecnología RISC.

Fig. 21. Arquitectura CISC



Elaborado por: Cristian Cuji.

CISC procesa grandes cantidades de información y a su vez controla varios dispositivos que se encuentran gestionados por el procesador, como por ejemplo los modelos Intel Pentium poseen varios núcleos como dual core, core 2 duo, i3, i5 etc.

RISC

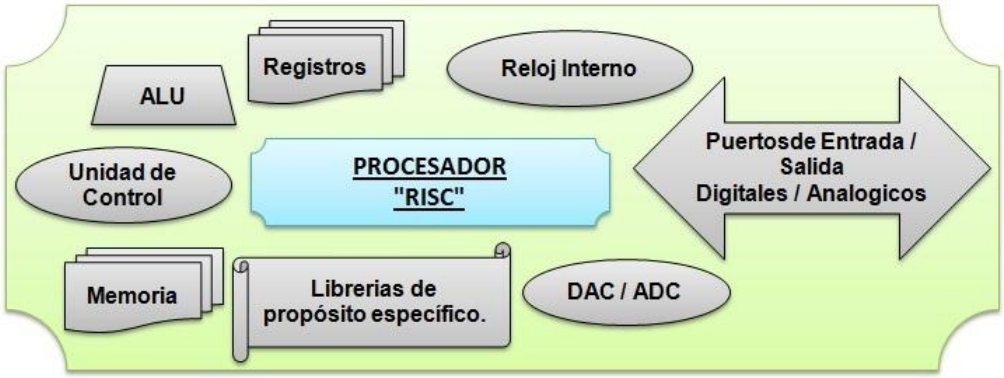
Reduced Instruction Set Computer o Conjunto de Instrucciones Reducidas de Computadora. El objetivo de esta arquitectura es obtener dispositivos con un número reducido de instrucciones evitando que el controlador o el procesador central ejecuten todas las instrucciones del sistema. Es una arquitectura de computadora relativamente

nueva, aunque fue desarrollada en 1964 por Seymour Cray su potencialidad y gran beneficio se dio a notar con el desarrollo de nuevas aplicaciones, entre ellas toda la gama de microcontroladores, su evolución dio como resultado los ARM (www.es.wikipedia.org, Wikipedia: La Enciclopedia Libre, 2012) que en la actualidad son utilizados en la elaboración de: PDAs, Apple iPods, Apple iPhone, iPod Touch y la mayoría de dispositivos de última generación. Ya para el 2010 el 90% de dispositivos electrónicos utilizaban por lo menos un dispositivo de tecnología RISC.

La filosofía de RISC es reducir el tamaño de procesos y el tamaño de la información, con dispositivos de propósito específico para un mejor procesamiento. Algunas características son:

- Instrucciones de tamaño fijo y presentado en un reducido número de formas.
- Solo las instrucciones de carga y almacenamiento acceden a la memoria de datos.
- Los dispositivos RISC disponen de núcleos y registros de propósito general.

Fig. 22. Arquitectura RISC

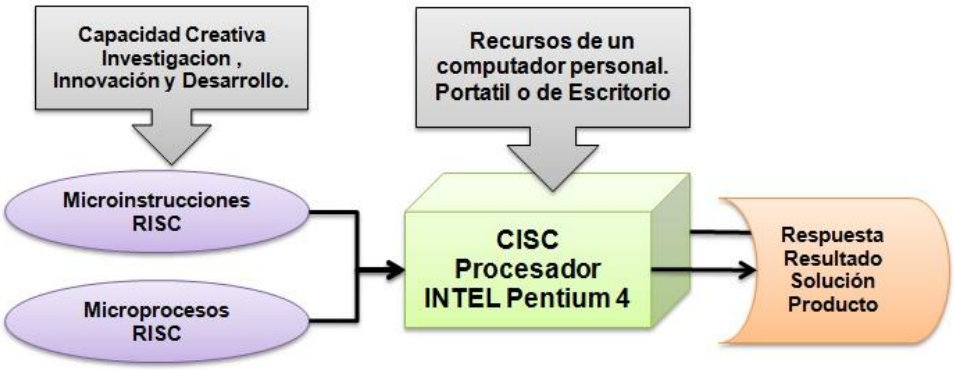


Elaborado por: Cristian Cuji.

2.3.2. Arquitectura predominante

El objetivo de RISC es reducir recursos y reducir el tamaño de procesos para obtener una respuesta. En la actualidad algunos sistemas CISC interactúan con microprocesos RISC. Es decir los microprocesos RISC entregan información ya procesada al sistema CISC, ahorrando recursos.

Fig. 23. Procesos RISC y CISC



Elaborado por: Cristian Cuji.

El Robotino Móvil Festo es un computador, internamente posee: CPU, Memoria, Registros, Puertos de entrada y salida, Controladores, etc. Implementar una pinza y brazo sin un control electrónico externo basado en arquitectura RISC, aumentaría la carga de información en el procesador del Robotino. Por medio de un Microcontrolador la información será previamente procesada y comunicada al Robotino ahorrando recursos en el procesador.

El módulo manipulador de objetos es un sistema o mecanismo de propósito específico, especialmente diseñado para realizar tarea de transporte y manipulación de objetos, será controlado por medio de un sistema RISC que a su vez es controlado por un sistema CISC que está representado por el computador del Robotino.

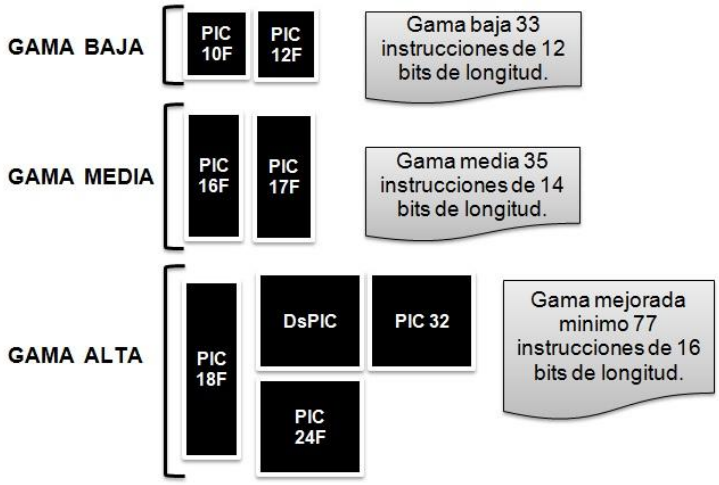
2.3.3. Microcontrolador

En el inicio los primeros microcontroladores cumplían funciones básicas, el lenguaje de programación era poco amigable, se requerían muchas líneas de código para ejecutar una acción y carecían en espacio de memoria. En la actualidad los lenguajes son mucho más amigables con el programador y la memoria del microcontrolador también se ha incrementado.

2.3.3.1. Gamas de microcontroladores PIC

Los Microcontroladores Microchip se pueden dividir en tres grandes familias o gamas, cada uno con sus respectivas características.

Fig. 24. Gamas de Dispositivos Microcontroladores Microchip Comerciales

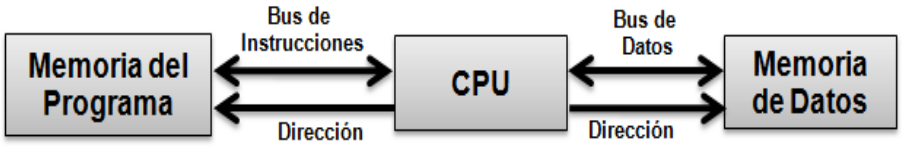


Fuente: (PREDKO, PROGRAMMING AND CUSTOMIZING THE PIC MICROCONTROLLER, 2002)
Elaborado por: Cristian Cuji.

En general todas las familias de cada microcontrolador incluyen en su interior las tres unidades funcionales principales de una computadora:

- Unidad central de procesamiento.
- Memoria.
- Periféricos de entrada y salida.

Fig. 25. Arquitectura Hardware dispone de dos memorias independientes



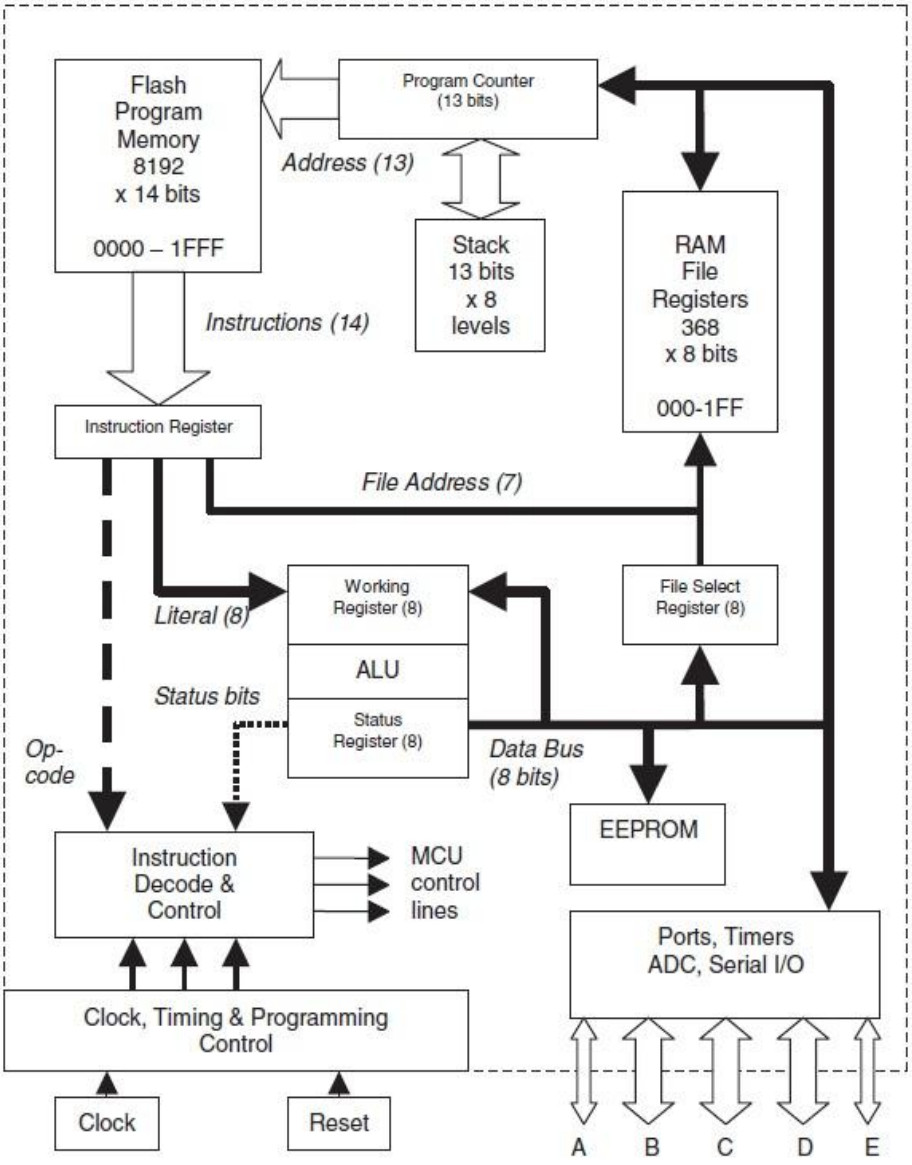
Fuente: (ANGULO USATEGUI & ROMERO, 2da edición 2006, pág. 2)
Elaborado por: Cristian Cuji.

Un microcontrolador cumple con las características antes mencionadas para ser un dispositivo de sistema RISC.

La información puede ser introducida a su interior por medio de los periféricos de entrada y salida, la información puede ser procesada en su interior en su unidad central de procesamiento, esta información puede servir para tomar decisiones de acuerdo a

instrucciones guardadas en su memoria y emitir una respuesta de nuevo por sus periféricos de entrada y salida.

Fig. 26. Diagrama de Bloques de la Ejecución de un Programa



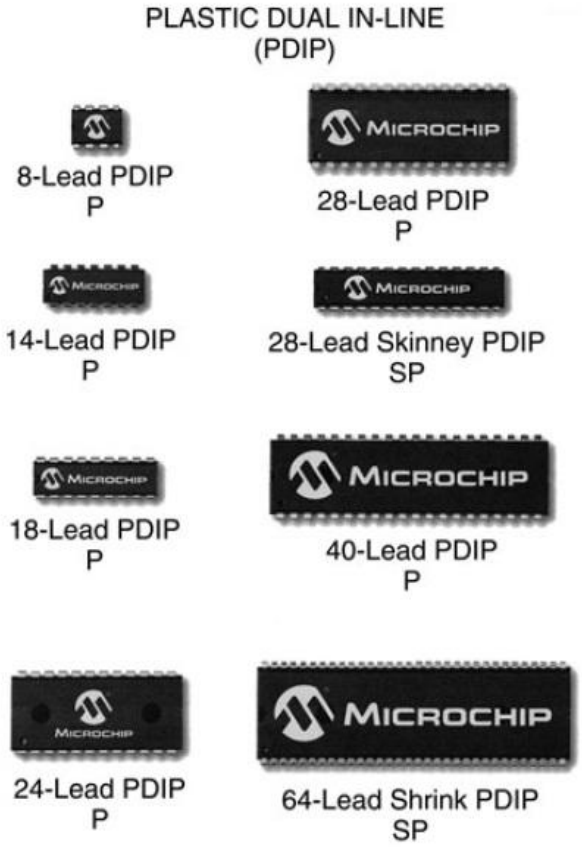
Fuente: (BATES, 2006)

Además de esto la tecnología actual de los microcontroladores puede establecer comunicaciones por medio de protocolos propios o establecer nuevas formas de comunicación con otros dispositivos, Ejemplo: Otros Microcontroladores, Computador, PLC, o diferentes tipos de maquinas.

2.3.3.2. Características de acuerdo a cada gama

La empresa MICROCHIP, comercializa diversas versiones de un mismo dispositivo, en algunos casos con pequeñas mejoras y en otros casos con mayores funciones. Los dispositivos operan con 8 bits de ancho de banda para datos y se encuentran en distinto tipo de encapsulado de 8 pines hasta 64 pines.

Fig. 27. Plastic dual in-line (DIP) OTP packages



Fuente: (PREDKO, PROGRAMMING AND CUSTOMIZING THE PIC MICROCONTROLLER, 2009)

Básicamente, todos los microcontroladores PIC ofrecen las siguientes características.

- Alrededor de 35 instrucciones RISC.
- Puertos de entrada / Salida
- Temporizadores internos de 8 bits, Watchdog y Interfaz de reloj externo.
- Power-on Reset y Ahorro de energía modo SLEEP.
- Modos de direccionamiento directo, indirectos y en relación.
- Memoria RAM de Datos y Memoria de programa EPROM o OTP

Adicionalmente algunos dispositivos ofrecen:

- Canales de entrada analógicos y Comparadores analógicos.
- Circuitos temporizadores adicionales.
- Datos de memoria EEPROM y Flash EEPROM de la memoria del programa.
- Interrupciones de temporizador externa.
- Circuito de programación y Oscilador interno.
- Interfaz serial USART.

Algunas características específicas de cada familia las revisaremos a continuación:

Familia PIC12Cxxx

Pic 12C508, fue de esta familia el más vendido gracias a su bajo costo, consta de 8 pines, es un dispositivo con 512x12 de memoria de programa EPROM y 25 bytes de memoria RAM de datos. El dispositivo puede funcionar hasta 4MHz con señal de reloj externo, posee 33 instrucciones. Y cuenta con 6 pines I/O, temporizador de 8 bits, power-on reset, watchdog timer, y un oscilador de 4 MHz RC internos.

Tabla- 2. Algunos PIC de Gama Baja

Microcontroller	Program Memory	Data RAM	Max Speed (MHz)	I/O Ports	A/D Converter
12C508	512 x 12	25	4	6	-
12C672	2048 x 14	128	10	6	4
12CE518	512 x 12	25	4	6	-
12CE673	1024 x 14	128	10	6	4
12CE674	2048 x 14	128	10	6	4

Fuente: (IBRAHIM, September, 2002)
Elaborado por: Cristian Cuji.

Familia PIC16CXXX

Los PIC mas comerciales de esta familia son PIC16C554, PIC16C54 poseen una arquitectura similar con instrucciones son de 14 bits de ancho. La memoria EPROM con 512 x 14 y la memoria de datos son de 80 bytes de RAM. Posee 13 pines de puertos de entrada / Salida, un temporizador watchdog. Algunos otros miembros de esta familia como el PIC16C71 incorpora cuatro canales de convertidor Análogo / Digital con 1024

x 14 memoria de programa EPROM, 36 bytes de datos de memoria RAM y el temporizador.

PIC16F877 es un microcontrolador sofisticado que ofrece ocho canales de convertidores Análogo / Digital de 8192 x 14 de memoria de programa, 368 bytes de memoria de datos, 33 pines Entrada / Salida, USART, la interfaz de bus I2C, SPI, bus de interfaz, 3 temporizadores, y temporizador interno.

Tabla- 3. Algunos PIC de Gama Media

Microcontroller	Program Memory	Data RAM	Max Speed (MHz)	I/O Ports	A/D Converter
PIC17C42	2048 x 16	232	33	33	-
16C554	512 x 14	80	20	13	-
16C64	2048 x 14	128	20	33	-
16C71	1024 x 14	68	20	13	4
16F877	8192 x 14	368	20	33	8
16F84	1024 x 14	68	10	13	-

Fuente: (IBRAHIM, September, 2002)
Elaborado por: Cristian Cuji.

Familia y PIC17CXXX PIC18CXXX

De los PIC más comercializados en este caso son, PIC17C42, este microcontrolador tiene un 2048 x 16 memoria de programa. La memoria de datos es 232 bytes. Además posee 33 pines Entrada / Salida, USART, 4 temporizadores, el temporizador de watchdog, 2 registros de captura de datos y salidas PWM.

PIC 17C44 es similar, pero ofrece más memoria de programación.

PIC18CXXX, los miembros de esta familia incluyen PIC18C242 con la memoria de programa 8192 x 16, además posee memoria de datos de 512 bytes ,23 pines entrada / salida, 5 canales de entradas y salidas análogo / digital de 10 bits de ancho, USART, I2C, SPI y las interfaces de bus, salidas PWM, 4 temporizadores, temporizador watchdog, comparadores, captura de registros y múltiples instrucciones.

Tabla- 4. Algunos PIC de Gama Alta

Microcontroller	Program Memory	Data RAM	Max Speed (MHz)	I/O Ports	A/D Converter
17C43	4096 x 16	454	33	33	-
17C752	8192 x 16	678	33	50	12
18C242	8192 x 16	512	40	23	5
18C252	16384 x 16	1536	40	23	5
18C452	16384 x 16	1536	40	34	8

Fuente: (IBRAHIM, September, 2002)
Elaborado por: Cristian Cuji.

En la actualidad un microcontrolador (MCU, μC o UC), es un dispositivo electrónico, con la capacidad de cumplir órdenes previamente programadas. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Mediante el lenguaje de programación los microcontroladores se han desarrollado desde funciones sencillas hasta funciones muy complejas según la necesidad. Los leguajes de programación pueden dividirse en dos grupos: Lenguajes de Alto nivel y Bajo nivel.

2.3.3.3. Lenguaje de programación de Bajo Nivel

La programación de bajo nivel proporciona un conjunto de instrucciones aritméticas y lógicas sin la capacidad de encapsular dichas instrucciones en funciones que no estén ya contempladas en la arquitectura del hardware. Este lenguaje es complicado para desarrollar programas, porque viene dado por las especificaciones técnicas del hardware, es decir muy cercano a lenguaje de máquina. (Grup Enciclopedia Catalana, 2010). El lenguaje ensamblador, o assembler (assembly language) es un lenguaje de programación de bajo nivel. Es útil para programar dispositivos como: memorias, microprocesadores, microcontroladores, y otros circuitos integrados programables. Implementa una representación simbólica de los códigos de máquina binarios y otras constantes necesarias para programar una arquitectura dada de CPU y constituye la representación más directa del código máquina específico para cada arquitectura.

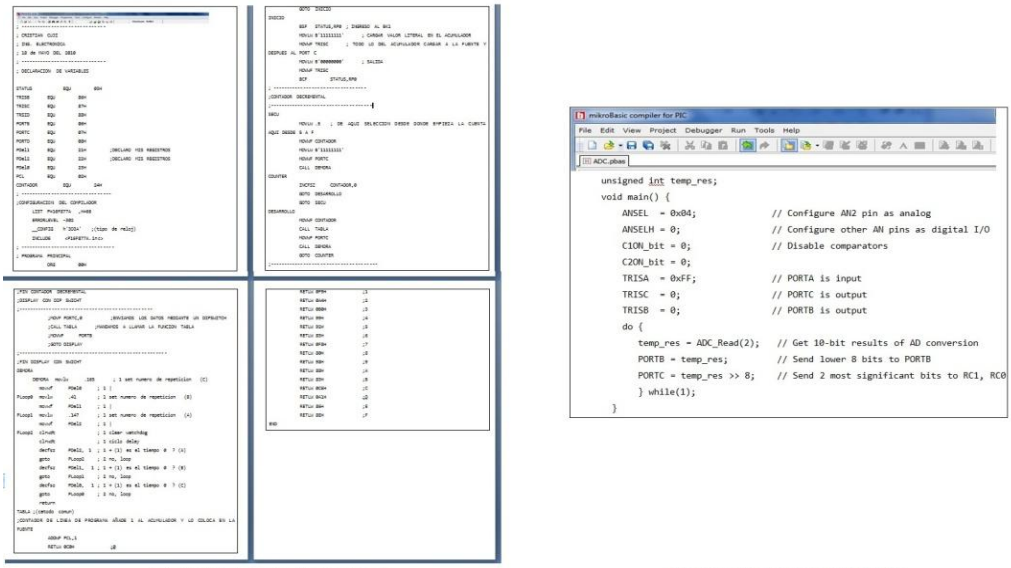
2.3.3.4. Lenguaje de programación de Alto Nivel

La programación de alto nivel se caracteriza por expresar los algoritmos de una manera menos abstracta y más adecuada a la capacidad cognitiva del ser humano. En los compiladores de alto nivel existen funciones, librerías e incluso ejemplos de conexión y

programación de microcontroladores. Los lenguajes de alto nivel requieren de ciertos conocimientos de programación para realizar las secuencias de instrucciones lógicas.

Haciendo una comparación entre los lenguajes de máquina y un lenguaje de alto nivel, vamos a encontrar marcadas diferencia. Los lenguajes de alto nivel incluyen librerías especializadas que facilitan la programación, mientras que en lenguaje ensamblador todo el programa debe ser desarrollado en su totalidad. Esta puede ser la principal ventaja del lenguaje de alto nivel con relación al lenguaje de bajo nivel, específicamente sobre aplicaciones que no requieran algoritmos muy complicados. Por ejemplo a continuación un programa de similares características, desarrollado en MPLAB (Lenguaje Ensamblador) y desarrollado en MikroBasic.

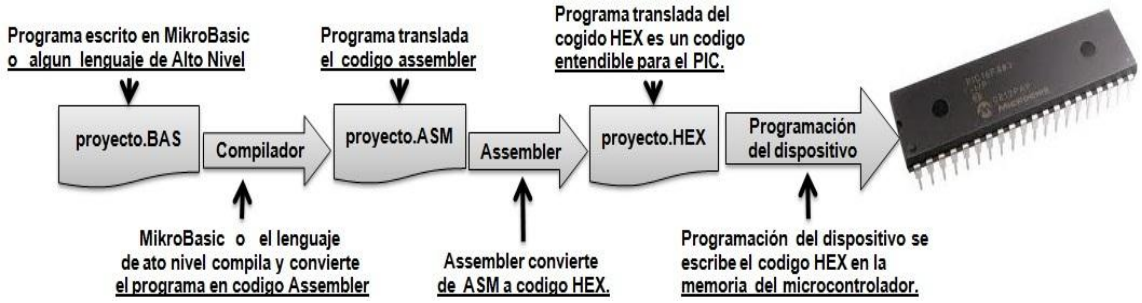
Fig. 28. Comparación entre Lenguaje Ensamblador y Lenguaje de alto nivel



Fuente: Software MPLAB y Mikrobasic
Elaborado por: Cristian Cuji.

La diferencia es muy clara en MikroBasic un lenguaje de alto nivel solo se necesita de 15 líneas de programación mientras que en lenguaje ensamblador se necesitan alrededor de 40 líneas para realizar un ADC (Codificador Análogo - Digital). En conclusión depende de la aplicación que se requiera se debe escoger un lenguaje u otro.

Fig. 29. Proceso de programación de un dispositivo Microcontrolador



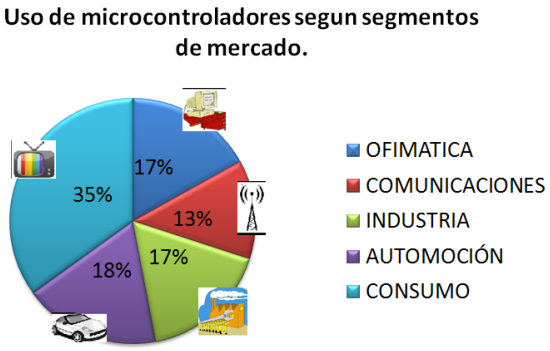
Fuente: (MATIC, BASIC FOR PIC MICROCONTROLLERS, January 2003)
Elaborado por: Cristian Cuji.

Programas como MicroBasic, MicroC, MicroCode permiten desarrollar un conjunto de instrucciones basado en Lenguaje C o Basic al ser compilado genera un archivo con extensión .ASM (Assembler Source by Microcontroller), es decir se traduce a lenguaje ensamblador y finalmente a partir del archivo .ASM se genera un archivo .HEX, que básicamente es un formato o lenguaje de maquina prácticamente 0 y 1 ordenados en código hexadecimal que interpreta el microcontrolador como instrucciones que se van a realizar.

2.3.3.5. Uso de los microcontroladores según segmentos de mercado

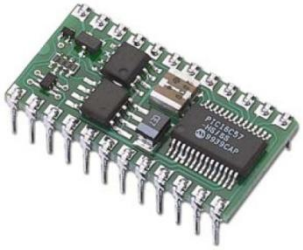
Las familias de los microcontroladores Pic son desarrollador por Microchip Technology Inc. En la actualidad algunos microcontroladores son más populares comercialmente, la venta de dispositivos supera los 120 millones de unidades cada año.

Tabla- 5. Distribución de ventas de micro controladores PIC en los principales segmentos del mercado



Fuente: (ANGULO USATEGUI & ROMERO, 2da edición 2006); Pág. 4
Elaborado por: Cristian Cuji.

Fig. 30. Microcontrolador de Montaje Superficial



Fuente: (LINDSAY, 2005)

2.4. Sensores y actuadores

2.4.1. Sensores

Según (DICCIONARIO DE LA LENGUA ESPAÑOLA 10maE, 2012): "m. Dispositivo que detecta una determinada acción externa, temperatura, presión, etc, y la transmite adecuadamente".

El sensor permite identificar un fenómeno físico, transformarlo en una señal eléctrica y enviar el dato obtenido al sistema de control. El sistema de control procesa la información proveniente del sensor y realiza una acción, enviando una nueva señal eléctrica hacia el actuador. Tanto sensores y actuadores se denominan como elementos finales de control.

Un sensor es un dispositivo que detecta el cambio de una variable física y lo traduce en una señal que puede ser procesada por un sistema electrónico. Al elemento activo de un sensor se le conoce como transductor. Los sistemas de monitoreo y control requieren sensores para medir cantidades físicas tales como posición, distancia, fuerza, deformación, temperatura, vibración y aceleración.

2.4.1.1. Interruptores de proximidad

Un sensor de proximidad consiste de un elemento que cambia su estado o una señal analógica cuando se acerca a un objeto, pero que con frecuencia no lo toca en realidad. Los métodos magnéticos, de capacitancia eléctrica, inductancia y de corriente parasita son particularmente adecuados para el diseño de un sensor de proximidad.

De los dispositivos más utilizados se pueden mencionar los interruptores, existen gran cantidad de diseños para los interruptores, entre ellos los de fin de carrera que pueden

ser de botón y los micro interruptores pulsadores, ambos sistemas se usan para abrir o cerrar conexiones dentro de un circuito.

Los interruptores se caracterizan por:

- (P) número de polos.- es un elemento móvil en el interruptor que establece o rompe conexiones.
- (T) número de tiros.- es un punto de contacto para un polo.
- (N.O.) Normalmente Abierto.
- (N.C.) Normalmente Cerrado.
- (SPTD) es un dispositivo de un solo polo.
- (SP) dispositivo de un solo tiro.
- (ST) dispositivo que abre o cierra una sola conexión.
- (SPDT) cambia el polo entre dos diferentes posiciones de tiro. ***

2.4.2. Actuadores

Son dispositivos eléctricos que transforman la energía eléctrica en una acción o generan un cambio de estado por medio de la energía eléctrica, neumática, hidráulica. Es decir un actuador generalmente transforma una señal eléctrica en movimiento, dicho movimiento se crea mediante una fuerza o momento de torsión que resulta en aceleración, desplazamiento lineal y angular, también se regulan o modulan la tasa y la potencia asociadas con el cambio. Por esta razón un aspecto importante en el diseño de un sistema electrónico - mecánico es la selección apropiada de actuadores.

2.4.3. Motores DC

"Adj. Que mueve.

m. Máquina destinada a producir movimiento a expensas de otra fuente de energía. Motor eléctrico, térmico, hidráulico" (DICCIONARIO DE LA LENGUA ESPAÑOLA 10maE, 2012)

Un motor eléctrico es un dispositivo electromecánico que emplean campos magnéticos para generar movimiento, mediante materiales ferromagnéticos se guía y concentra estos campos. Debido a que la permeabilidad magnética de los materiales ferromagnéticos es alta, de hasta diez mil veces el espacio que lo rodea. El electromagnetismo es una rama de la física que estudia los fenómenos eléctricos y magnéticos en una sola teoría, cuyos fundamentos fueron sentados por Michael Faraday y formulados por primera vez de modo completo por James Clerk Maxwell.

Los motores de corriente directa (DC) existen en una gran cantidad de diseños debido a las características de torque y velocidad que se pueden lograr con diferentes configuraciones eléctricas. La velocidad del motor DC pueden ser controlar con suavidad y en la mayoría de casos el giro de motor puede ser reversible. Poseen una rápida respuesta gracias a que tiene movimiento de inercia del rotor. Los motores de DC se clasifican en cuatro grandes categorías de acuerdo con la manera en que se crean los campos magnéticos del estator:

Motor Imán permanente (PM).- Los campos del estator son proporcionados mediante imanes permanentes, que no requieren fuente de poder externa y por lo tanto no producen calentamiento I^2R. Un motor PM es más ligero y más pequeño que otros motores DC equivalentes porque la intensidad del campo del imán permanente es alta, son fáciles de invertir al conmutar la dirección del voltaje aplicado, pues la corriente y el campo cambian de dirección solo en el rotor. Los Motores DC PM pueden ser motores con escobillas, sin escobillas o de pasos.

Devanado Shunt.- Tienen armadura y devanados de campo conectados en paralelo, que se activan mediante la misma fuente. La corriente de carga total es la suma de las corrientes de armadura y campo. Los motores Shunt muestran velocidad casi constante sobre un gran rango de carga, tienen torque de arranque de aproximadamente 1.5 veces el torque operativo nominal.

Devanado en serie.- Tiene devanados de armadura y campo conectados en serie, de modo que las corrientes de armadura y campo son iguales. Los motores en serie muestran torques de arranque muy altos, velocidad enormemente variable dependiente de la carga, y la velocidad muy alta cuando la carga es pequeña.

Devanado compuesto.- Parte de la corriente de carga pasa a través de los devanados en derivación. La velocidad máxima de un motor compuesto es limitada a diferencia de un motor en serie, pero su regulación de velocidad no es tan buena como con un motor en derivación.

2.4.4. Servomotores

Una de las aplicaciones de los Motores DC de Imán permanente son los servomotores, que buscan ganar una mayor fuerza o mayor torque. Cuando se usa un motor en una aplicación de control de posición o velocidad con retroalimentación de sensor a un controlador, se denomina como servomotor. Los motores PM se usan solo en aplicaciones de baja potencia pues su potencia nominal usualmente se limita a 5 hp (3278 W).

Están constituidos esencialmente por dos partes:

- **Estator**: parte fija construida a base de cavidades en las que van depositadas las bobinas.

- **Rotor**: parte móvil construida mediante un imán permanente. Este conjunto va montado sobre un eje soportado por dos cojinetes que le permiten girar libremente.

La precisión y repetitividad que presentan esta clase de motores lo habilitan para trabajar en sistemas abiertos sin realimentación.

2.4.5. Motor a Pasos

Un motor paso a paso, como todo motor, es en esencia un convertidor electromecánico, que transforma energía eléctrica en mecánica. Mientras que un motor convencional gira libremente al aplicarle una tensión, el motor paso a paso gira un determinado ángulo de

forma incremental (transforma impulsos eléctricos en movimientos de giro controlados), lo que le permite realizar desplazamientos angulares fijos muy precisos (pueden variar desde 1,80º hasta unos 90º).

La extensa disponibilidad y el bajo costo de la microelectrónica y la electrónica de potencia han hecho que en una amplia variedad de aplicaciones, la máquina de reluctancia variable se vuelva competitiva con otras tecnologías de motores.

Con la excitación en secuencia de las fases de una máquina de reluctancia variable el rotor girará a pasos o de manera gradual, rotando a través de un ángulo especifico por paso. Los motores de pasos están diseñados para sacar provecho de esta característica. Los motores con tales características con frecuencia combinan el uso de una geometría de reluctancia variable con imanes permanentes para incrementar el par y la precisión de la posición.

Un motor a pasos a menudo se utiliza en sistemas de control digital donde el motor recibe comandos de bucle abierto en la forma de un tren de pulsos para hacer girar un eje o mover un objeto a una distancia específica. Algunas aplicaciones típicas incluyen los motores posicionadores de cabezas de impresión, mando en unidades de disco y reproductores de discos compactos, en mesas de trabajo los motores de posicionamiento de herramientas en equipo de maquinado numérico controlado. En muchas aplicaciones es posible obtener información de posición simplemente con llevar la cuenta de los pulsos enviados al motor, en cuyo caso no se requiere sensores de posición ni control de retroalimentación.

2.5. Resumen del Capítulo

El primer capítulo fue orientado al estudio de los principales temas del desarrollo del proyecto: Robótica, Electrónica y Mecánica. Se describió un resumen de la historia, características y situación actual de la tecnología, además del uso de conocimientos y herramientas que se van a ser utilizadas en el desarrollo del proyecto.

En la historia de la robótica se revisó la evolución desde los primeros sistemas de movimiento autónomo, hasta avanzados sistemas de actualidad tomando un interés particular en el Robotino de Festo, que es pilar fundamental en el desarrollo de esta tesis, donde se aclara que el Robotino es un robot didáctico, pero con características que lo vuelven apropiado para la investigación de estudiantes y para el desarrollo de nuevas aplicaciones. Se dedicó también parte de la investigación a conocer en el mercado existente de Robots y estadísticas de ventas a nivel mundial.

Al ser un trabajo de Desarrollo e Implantación, es necesario tener un mínimo conocimiento de los materiales de construcción mecánica, dicho tema fue desarrollado en el subtema Mecánica, donde se revisó los diferentes tipos de materiales metálicos, la evolución de herramientas y el uso de un lenguaje universal el lenguaje gráfico, dibujo técnico y Auto CAD.

Por último se realizó un estudio de las dos arquitecturas actuales y predominantes en los sistemas de control: CISC y RISC. Aquí se acordó que el uso de microcontroladores implica la aplicación de tecnología RISC. Para tener presente se revisó tipos y características principales de estos dispositivos y su uso en la actualidad, reflejado en estadísticas de ventas a nivel mundial.

La Electrónica, Mecánica y Robótica son los tres lineamientos que van a llevar a cabo la implementación del proyecto. El siguiente capítulo presenta la descripción del diseño, desarrollo e implementación del sistema.

CAPÍTULO 3
DISEÑO, DESARROLLO E IMPLEMENTACIÓN

Las principales actividades desarrolladas en el proyecto, fueron principalmente:

- Diseñar el Sistema Mecánico.
- Diseñar el Sistema Electrónico / Eléctrico
- Desarrollar el Software de control.
- Implementar el sistema en el Robotino.

La primera actividad es elaborar un modelo mecánicamente funcional, tomando en cuenta la compatibilidad con un sistema de control electrónico basado en un microcontrolador PIC para finalizar con el desarrollo del software de control, tanto en la tarjeta electrónica y en el Robotino. Todo esto con el objetivo de desarrollar un *Sistema de Transporte Guiado Automático*. (FESTO Didactic GmbH & Co. KG, 2008)

Los cuatro lineamientos serán explicados y desarrollados en mayor detalle de manera individual a continuación.

3.1. Diseño mecánico

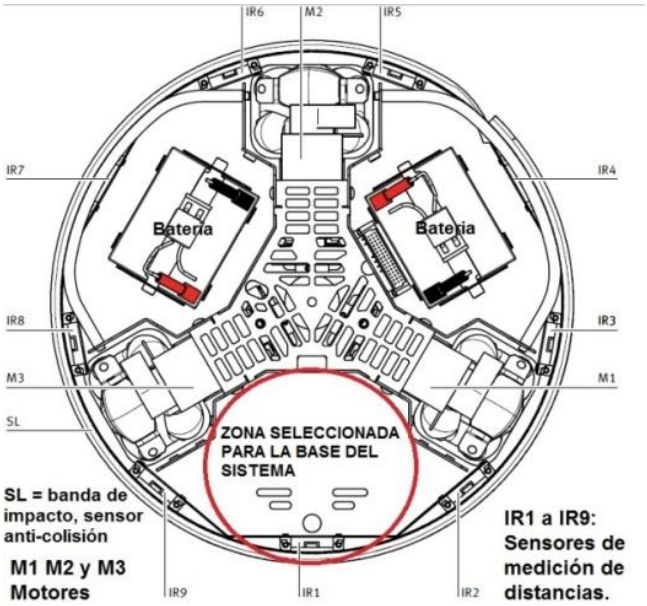
El diseño mecánico es la construcción de un sistema de cuerpo rígido, con el objetivo de transmitir un movimiento circular ejecutado por un motor de pasos hacia un tornillo sin fin generando un movimiento lineal en sentido vertical. El sistema contará con una pinza mecánica diseñada para abrirse y cerrarse. Además se debe considerar que el sistema mecánico sea compatible, adaptable y estéticamente aceptable al sistema Robotino de Festo.

Elaborar un gráfico inicial del proyecto a desarrollarse ayuda para empezar con la construcción del armazón mecánico, se debe recalcar que para el desarrollo del sistema mecánico se conto con la ayuda y asesoría de un Mecánico Industrial durante avance del proyecto. Para esto es necesario realizar un bosquejo sin mayor detalle pero con medidas exactas.

3.1.1. Descripción del diseño mecánico

El objetivo es realizar un bosquejo o diseño del armazón mecánico del sistema manipulador de objetos y su ubicación en el Robotino. Para esto Robotino cuenta en su chasis con un espacio muy amplio en su frente, generalmente en este lugar se ubica la cámara web y fue seleccionado como el lugar apropiado para ubicar la base del módulo.

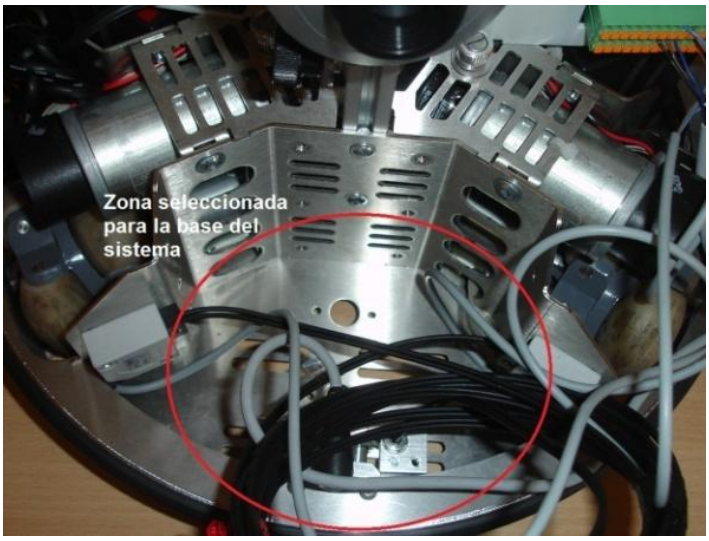
Fig. 31. Chasis del Robotino



Fuente: (FESTO B. W., 2007)
Elaborado por: Cristian Cuji.

El chasis del Robotino está conformado por tres áreas muy definas, que se muestran en la Fig. 31. Chasis del Robotino Dos áreas están ocupadas por las baterías y un tercio se encuentra vacio y precisamente esta ubicado en el frente del Robotino, en este lugar se colocará la base del sistema manipulador de objetos.

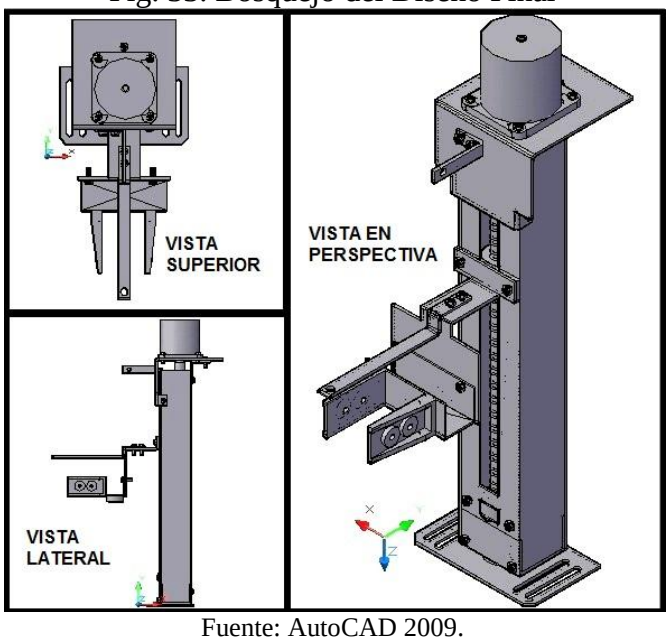
Fig. 32. Base para el soporte del sistema mecánico



Elaborado por: Cristian Cuji.

La principal idea es desarrollar un sistema con la capacidad de transmitir el movimiento desde el motor de pasos hacia un tornillo sin fin ubicado en el centro del tubo cuadrado, transformando un movimiento circular desde el motor, en movimiento lineal hacia la pinza. A continuación se puede visualizar el grafico del diseño final del Sistema Manipulador.

Fig. 33. Bosquejo del Diseño Final



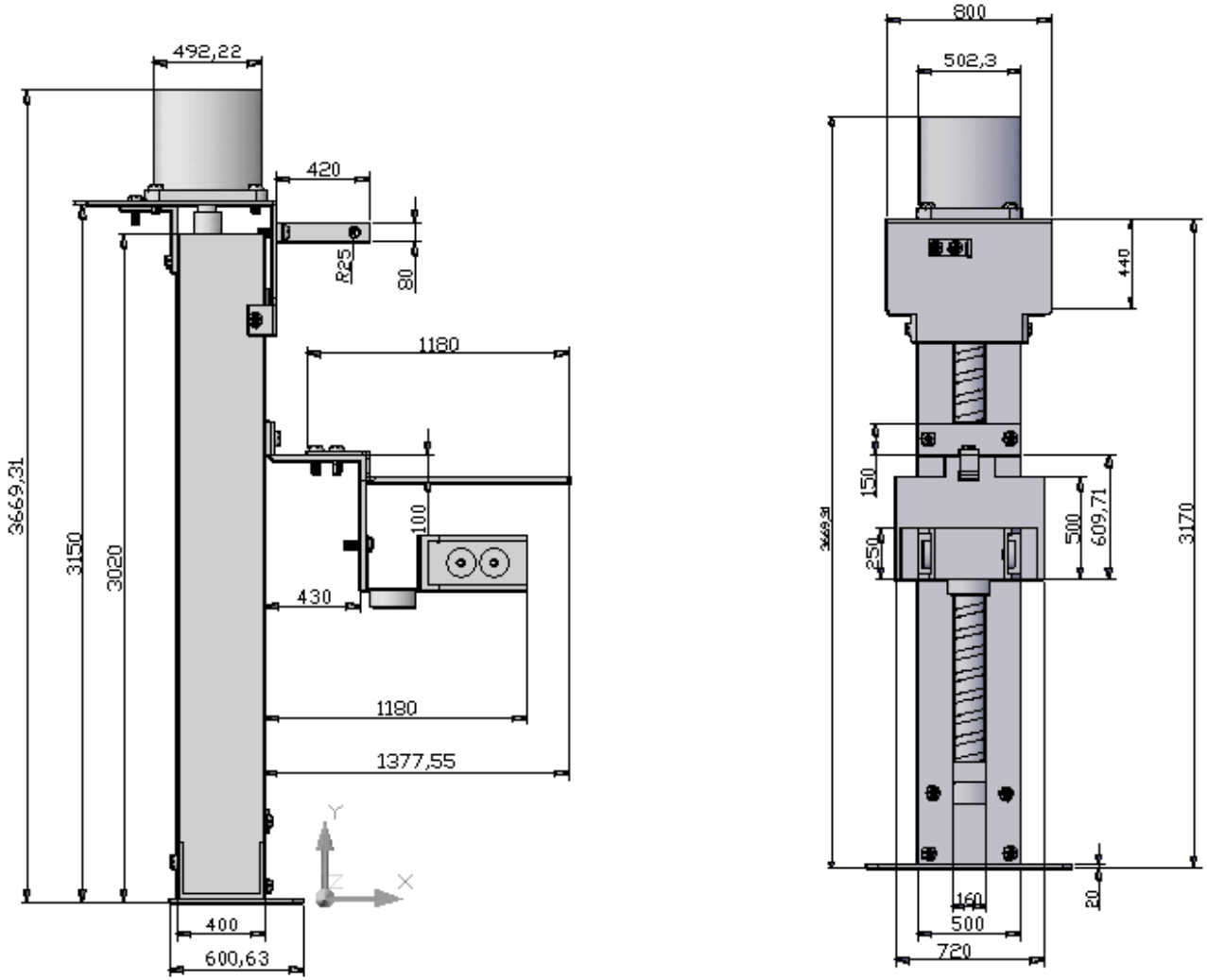
Fuente: AutoCAD 2009.
Elaborado por: Cristian Cuji.

Los principales elementos del sistema mecánico son:

- La Base.
- Tubo Cuadrado o Guía.
- Tornillo sin Fin.
- Brazo extendido.
- Pinza o Gripper.
- Motor de pasos.
- Sistema de engranes.

Tomando como referencia el diseño realizado en Auto CAD se procedió a elaborar el Sistema Mecánico,

Fig. 34. Dimensiones del Sistema Mecánico



VISTA LATERAL **VISTA FRONTAL**

Fuente: AutoCAD 2009 (*Valores en escala de x100 (Ej: 1180/100 =11.8 cm).)
Elaborado por: Cristian Cuji.

3.1.2. Resultado del diseño mecánico

Algunos elementos y materiales utilizados para el desarrollo del proyecto fueron:

Tabla- 6. Principales elementos mecánicos

Principales elementos mecánicos.			
Ítem	**Elemento**	**Cantidad**	**Medida**
1	Tubo cuadrado de Aluminio	1	4cm x 3cm x 2mm 30cm
2	Platina de aluminio	1	10cm x 100cm x 2mm
3	Tornillos y turcas	30	1/8 pulg
4	Tornillo sin fin de hierro dulce	1	25 cm
5	Rulimanes	2	

Elaborado por: Cristian Cuji.

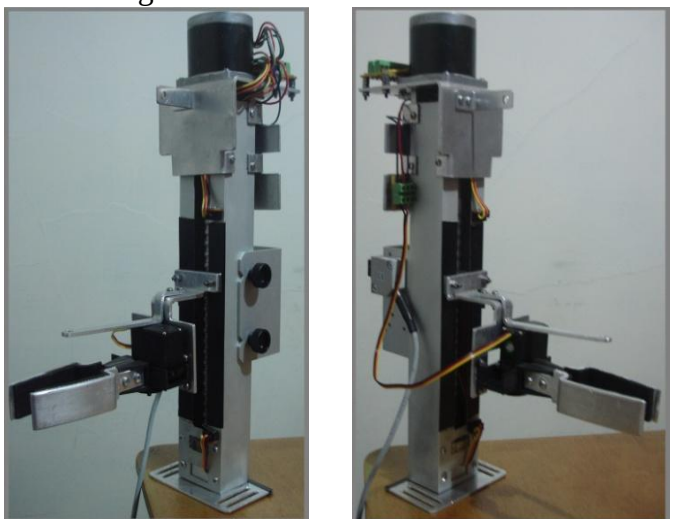
Tabla- 7. Características del Tornillo sin fin

Características del Tornillo sin fin.		
1	Material	Acero de transmisión
2	Rosca	Diametral - 18
3	Paso	3 hilos / Pulgada
4	Diámetro	14 mm.

Elaborado por: Cristian Cuji.

El resultado final se muestra a continuación en la siguiente figura.

Fig. 35. Estructura Mecánica Final



Elaborado por: Cristian Cuji.

3.2. Diseño electrónico

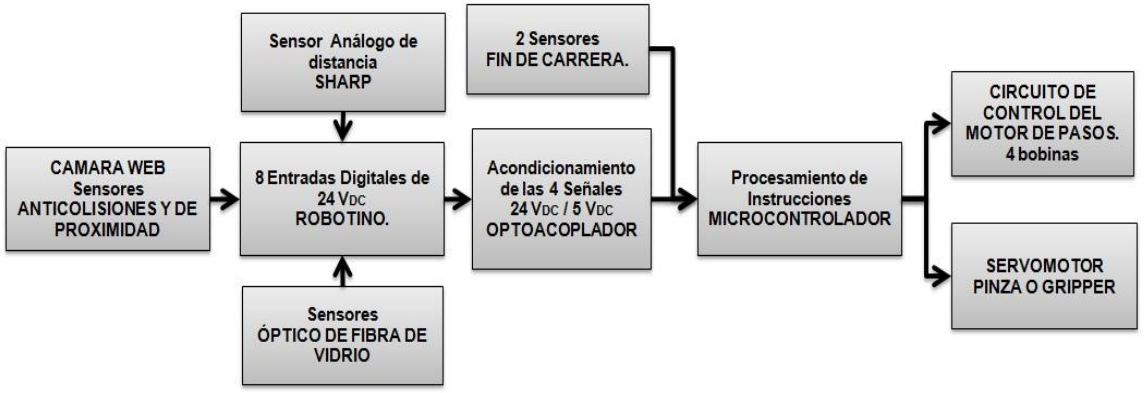
El diseño electrónico consiste en el desarrollo y la implementación de un sistema de control electrónico modular al Robot Móvil Robotino, el sistema es una interfaz que permite la comunicación del módulo manipulador de objetos hacia el Robotino y

viceversa, mediante el uso de dispositivos electrónicos entre ellos un Microcontrolador Microchip. Una vez desarrollado la herramienta mecánica, el paso siguiente es controlar el sistema o módulo manipulador, para esto es necesario entender el funcionamiento del sistema y desarrollar el tipo de control, además seleccionar los dispositivos electrónicos adecuados.

Uno de los objetivos principales es controlar la corriente de manera eficiente, para enviar la máxima corriente posible hacia las bobinas del motor generando la mayor cantidad de torque posible desde el motor al tornillo sin fin.

3.2.1. Descripción del funcionamiento

Fig. 36. Diagrama de bloques del funcionamiento del circuito de control



Elaborado por: Cristian Cuji.

Robotino es un sistema robot móvil, prácticamente es un computador que funciona con sistema operativo Linux, está provisto de una cámara web y sensores entre los que se puede mencionar, los sensores anticolisión, sensores de proximidad y sensores ópticos. Además consta de un módulo de entradas / salidas digitales y analógicas.

El objetivo es desarrollar una tarjeta con la capacidad de interpretar las señales digitales provenientes del Robotino procesarlas y generar una respuesta. Entonces es necesaria la elaboración de un sistema electrónico basado en una tarjeta de control y adquisición de datos, acondicionando los voltajes provenientes del Robotino para que sean adaptables a las entradas del microcontrolador. Un Microcontrolador MICROCHIP de gama alta es el

responsable de procesar las señales provenientes del Robotino y de los sensores finales de carrera directamente conectados, para controlar al motor de pasos y al servo motor.\

Al observar el sistema como un proceso los sensores, cámara web y las demás señales que emite el Robotino son consideradas como las entradas del sistema de control electrónico. La tarjeta de adquisición y control de datos es la encargada de generar el proceso de control, interpretar las entradas procesar la información y generar una respuesta o resultado. El resultado es controlar el motor de pasos y el servomotor.

3.2.2. Circuito de control

El sistema electrónico funciona con una fuente externa de +12 V_{DC} a 1.3Amp. Recibe cuatro bits de instrucciones o señales digitales de +24 V_{DC} desde el Robotino y recibe la señal de dos sensores fin de carrera. De acuerdo a las instrucciones recibidas, el Microcontrolador posee un set o conjunto de instrucciones reducidas (RISC) capaces de interpretar las instrucciones y ejecutar acciones, estas son:

- Generar una secuencia para el control de un motor de pasos.
- Inversión del giro del motor de pasos.
- Ubicación en posiciones especificas en el movimiento lineal de la pinza.
- Control del servo motor encargado de abrir y cerrar la pinza.

La tarjeta de control básicamente posee los siguientes elementos:

- PIC 18F2550
- Cristal de 12000 Hz
- Transistor TIP 122.
- Opto acoplador PC 817.
- Regulador de voltaje 7805.
- Diodos rectificadores.
- Diferentes valores de resistencias.

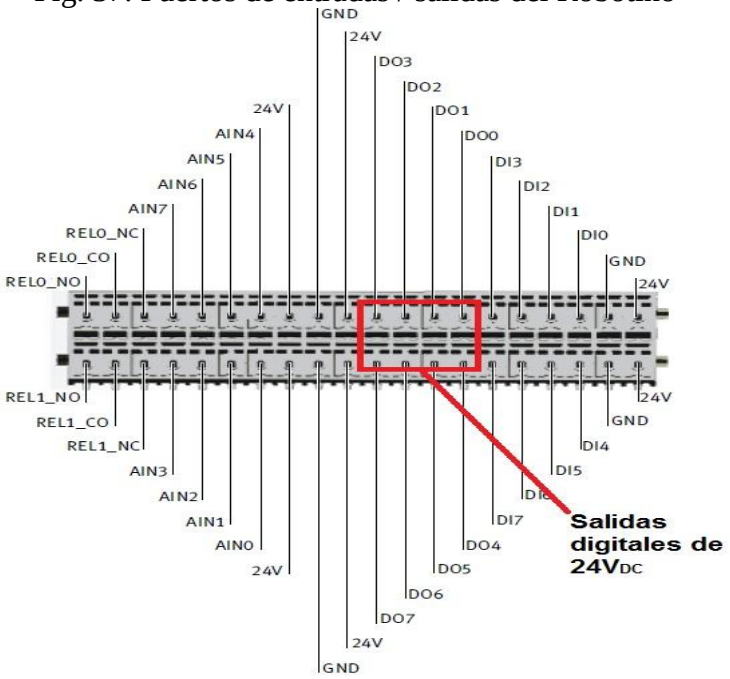
3.2.2.1. Salidas digitales del robotino

El Robotino está incorporado de un conjunto de borneras, que posee:

- 8 entradas analógicas (0 a 10 V) (AIN0 hasta AIN7)
- 8 entradas digitales (DI0 hasta DI7)
- 8 salidas digitales (DO0 hasta DO7) (UTILIZADAS EN EL PROYECTO)
- 2 relés para actuadores adicionales (REL0 y REL1). Los contactos de los relés pueden utilizarse como NA, NC o conmutados.

Las salidas del Robotino generan un voltaje de $24V_{DC}$ a 300 mA, por lo cual es necesario acondicionar la señal digital de $24V_{DC}$ a $5V_{DC}$, que es un valor de voltaje aceptado por el microcontrolador. Para conseguir esto es necesario el uso de opto acopladores, una de las ventajas de usar opto acopladores es evitar posibles cortos circuitos que se generen en la tarjeta de control y evitar un daño al Robotino.

Fig. 37. Puertos de entradas / salidas del Robotino



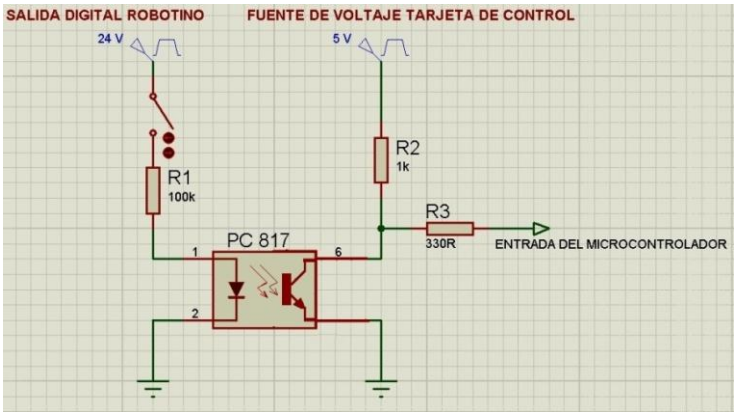
Fuente: (FESTO B. W., 2007)
Elaborado por: Cristian Cuji.

3.2.2.2. Acondicionamiento de voltaje

El siguiente circuito básicamente es necesario para acondicionar las señales digitales del Robotino hacia el microcontrolador por medio de los opto acopladores. La salida digital de 24 V_{DC} proveniente del Robotino está siendo simulada por el switch, el diodo se activa y acciona el transistor interno del opto acoplador obteniendo del otro lado un

voltaje de 5 V_{DC}, provenientes de una fuente externa de voltaje (Batería 12VDC 1.3 Amp.)

Fig. 38.Circuito de acondicionamiento desde el Robotino al microcontrolador



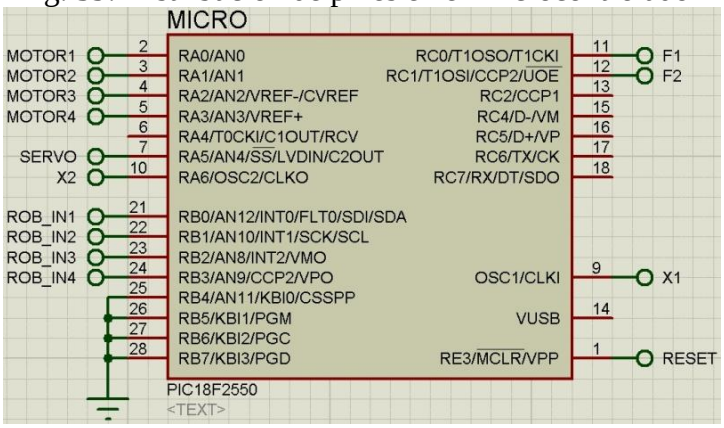
Fuente: Proteus - ISIS 7
Elaborado por: Cristian Cuji.

3.2.2.3. Distribución de puertos en el Microcontrolador

La distribución de pines del microcontrolador es la siguiente:

- Puerto B: ocho entradas digitales de 5 V_{DC} provenientes de las salidas digitales del Robotino de 24 V_{DC}, previamente acondicionadas. (Pines 21 al 28)
- Puerto A: desde RA0 a RA3 generan los pulsos para el control en el motor paso a paso y RA5 es el pin de control del servomotor. (Pines 2 al 5 y Pin 7)
- Puerto C: RC0 sensor fin de carrera alto, RC1 sensor fin de carrera bajo. (Pines 11 y 12) 0983152447

Fig. 39. Distribución de pines en el Microcontrolador



Fuente: Proteus - ISIS 7
Elaborado por: Cristian Cuji.

3.2.2.4. Control del motor de pasos y servomotor

Servomotor

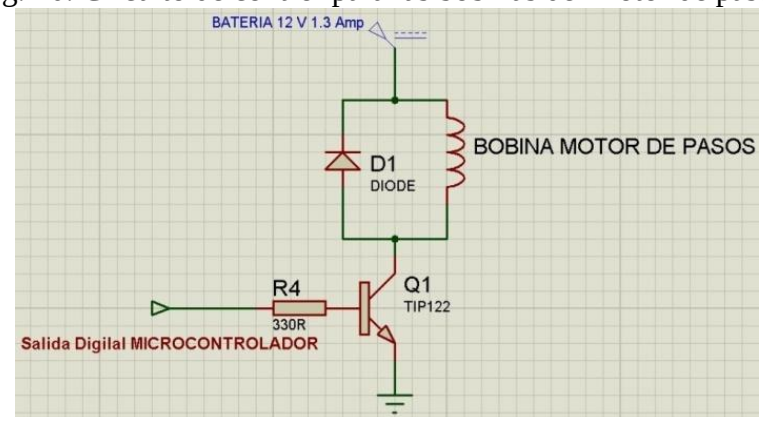
El servomotor se conecta directamente al microcontrolador de la siguiente manera:

- Cable rojo: + 5 VDC
- Cable negro: Tierra
- Cable Amarillo: Pin de Control (Pin7 RA5)

Motor Paso a Paso

Para el control del Motor de pasos es necesario aplicar una gran corriente que genere la mayor fuerza posible durante el giro, por esta razón se utiliza un transistor en modo de interruptor.

Fig. 40. Circuito de control para las bobinas del motor de pasos



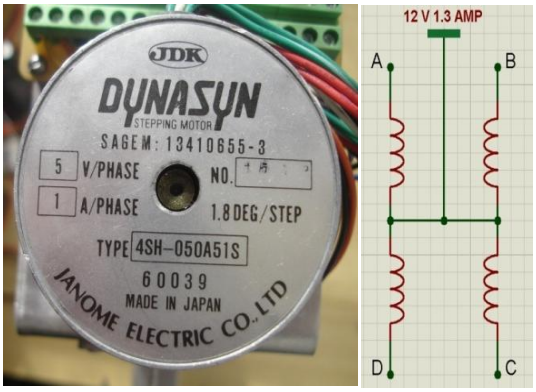
Fuente: Proteus - ISIS 7
Elaborado por: Cristian Cuji.

El microcontrolador genera el pulso de control que se aplicará a la base del transistor, esto produce la activación o desactivación entre colector y emisor, saturando al transistor y ubicándolo en la zona de conducción, lo que produce que las bobinas del motor de paso se activen o desactiven.

Cuando se trabaja con bobinas, es casi seguro obtener efectos inductivos de carga y descarga, que pueden llegar a ser perjudiciales porque generar un alto voltaje y corriente en sentido reverso, lo que produciría efectos térmicos (alta temperatura) en el transistor, la bobina y la fuente de voltaje.

Una solución es usar un diodo en anti paralelo a la bobina que encierra el voltaje y la corriente, evitando el efecto de descarga en sentido reverso, este diodo recibe el nombre de Diodo Clamp y es utilizado en el circuito para garantizar el buen funcionamiento del motor, evitando los efectos térmicos en otras palabras evitando que el motor se caliente y protegiendo a la fuente de voltaje.

Tabla- 8. Características del motor de Pasos

Características del Motor de Pasos.		
MARCA	DYNASYN, JDK Janome Electric CO. LTD	
TYPE	4SH - 050A51S	
V / Phase	5V - 24V	
A / Phase	1 - 2 Amp	
PASOS	1.8 Deg/Step	
N° Bobinas	4 bobinas	
Color	Negro	

Elaborado por: Cristian Cuji.

Descripción del motor de pasos:

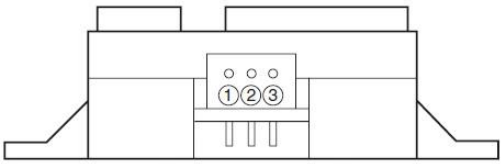
El motor de pasos fue seleccionado para el proyecto por sus características de fuerza y robustez, ya que se puede inducir un voltaje de entre 5V a 24V con una corriente de 1Amp a 2 Amp. Sin que se produzca un calentamiento excesivo o daño del motor. Posee cuatro bobinas, conectadas en pares de dos cada una por medio de dos bobinas comunes.

Es decir el motor presenta 6 cables de conexión. Blanco y Negro son cables comunes para las bobinas. Las bobinas están representadas con cables de colores: Verde, Rojo, Verde-Blanco y Rojo-Blanco.

3.2.2.5. Sensor de posicionamiento

Este bloque de la tarjeta electrónica es la encargada de acoplar un sensor Sharp Análogo de distancia, con el objetivo de controlar la posición o altura del gripper desde el Robotino.

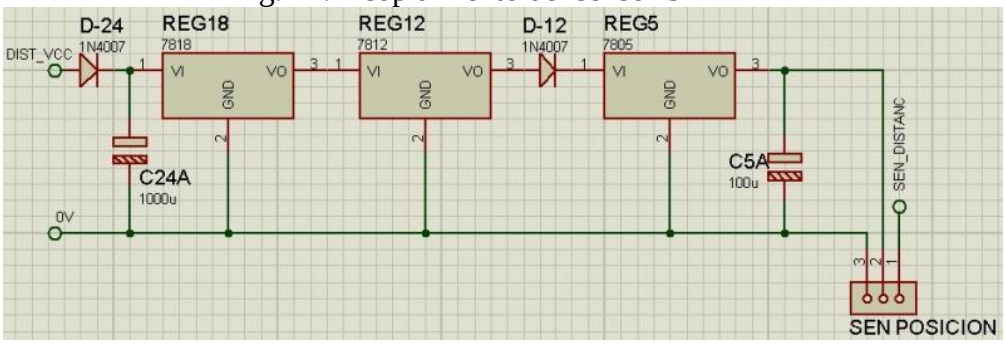
Tabla- 9. Características del sensor de posicionamiento

Características del sensor Sharp		
Marca	SHARP	
Descripción	Sensor ultrasónico de distancia.	
Modelo	GP2D120	
Vin	5V	
Vout	0,3V - 3,5V	
Distancia	4cm a 30cm	

Fuente: Data sheet Sharp GP2D120
Elaborado por: Cristian Cuji.

La etapa consiste en reducir el voltaje de $24V_{DC}$ a $5V_{DC}$, por medio de reguladores de voltaje de la gama 78xx. La señal del dato de la posición es recibida por el Robotino que es el que controla la altura de la posición del gripper según se requiera.

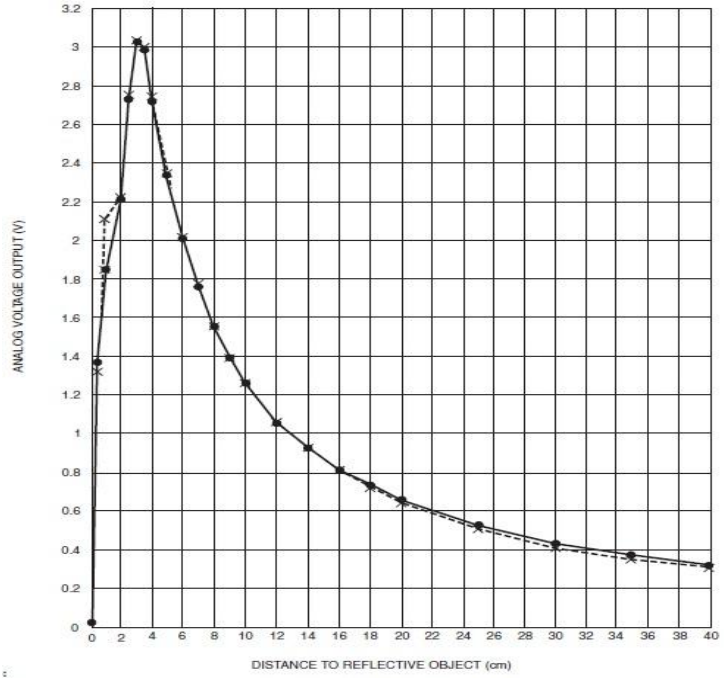
Fig. 41. Acoplamiento del sensor SHARP



Fuente: Proteus - ISIS 7
Elaborado por: Cristian Cuji.

El sensor seleccionado presenta la siguiente curva de funcionamiento de acuerdo a la posición en cm versus señal de voltaje, se debe considerar estos valores para la programación de posiciones.

Fig. 42. Curva de funcionamiento del Sensor SHARP



Fuente: Data sheet Sharp GP2D120

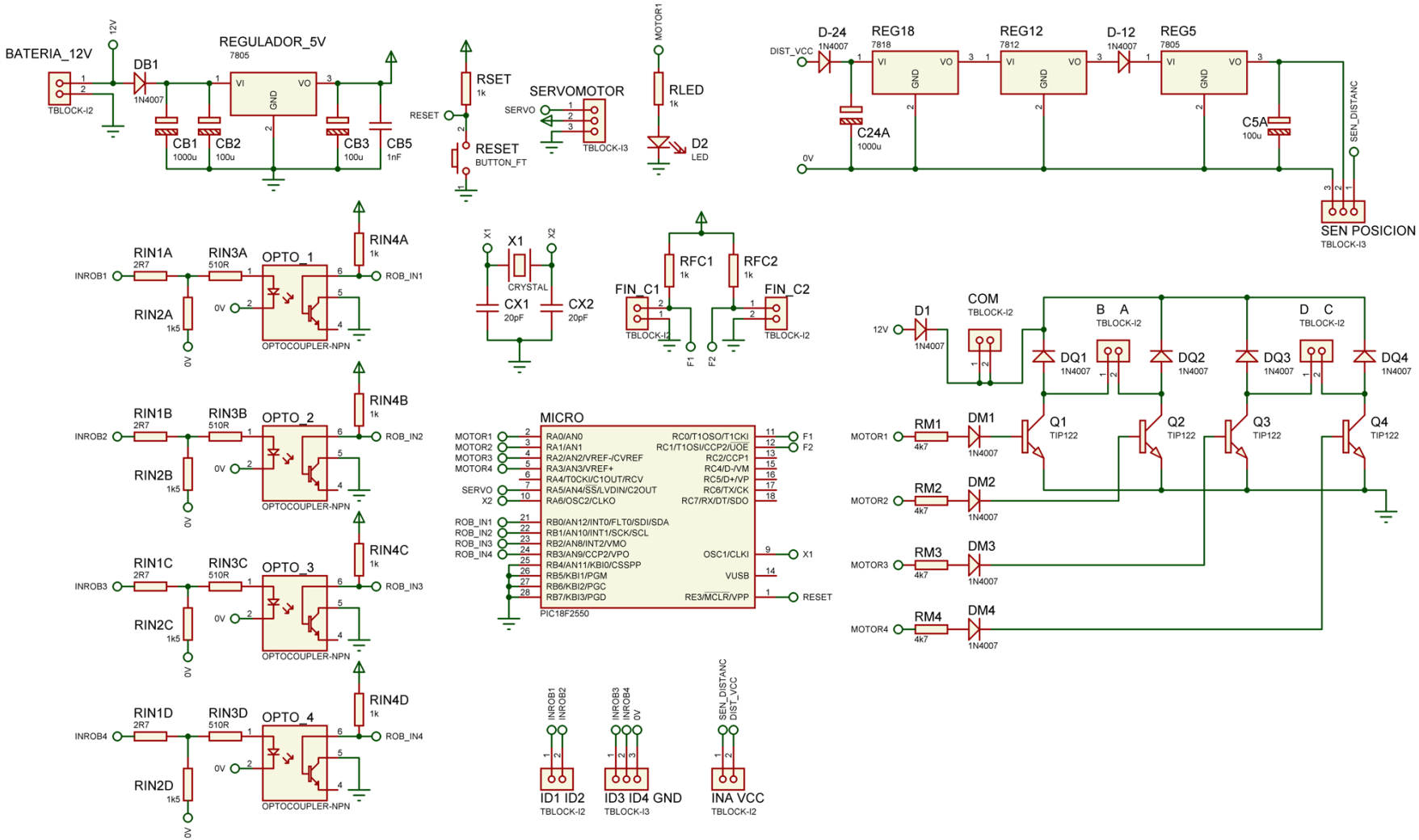
Elaborado por: Cristian Cuji.

3.2.2.6. Diagrama esquemático de la tarjeta de control

La tarjeta de control recibe las señales de $24V_{DC}$ desde el puerto de entradas y salidas del Robotino, estos voltajes son acondicionados por medio de un circuito de optoacoplamiento y reducidos de $24V_{DC}$ a $5V_{DC.}$ Los optoacopladores dividen las fuentes por un lado la fuente del Robotino y por otro lado, para el circuito de control y de potencia se utiliza una batería de $12V_{DC}$. La batería $12V_{DC}$ alimenta el microcontrolador y el motor de pasos. Se reduce el voltaje de $12V_{DC}$ a $5V_{DC}$ por medio de un regulador LM7805. Y finalmente se controla el voltaje y la corriente que ingresa al motor de pasos por medio de Transistores Darlington TIP122.

Se protegen a los elementos electrónicos y a la batería del sistema de corrientes reversas producidas por las bobinas del motor de paso utilizando diodos 1N4007, además de un capacitor en la fuente del Microcontrolador para sostener el voltaje constante. A continuación se muestra el diagrama esquemático de la tarjeta de control:

Fig. 43. Diagrama Esquemático de la Tarjeta de Control



Fuente: Proteus - ISIS 7
Elaborado por: Cristian Cuji.

Con la ayuda del Software ARES de Proteus, por medio de circuito esquemático se puede obtener el Diagrama de Circuito Impreso, que se presenta a continuación en tamaño real.

Fig. 44. Circuito Impreso o Pistas. (Bottom Copper)

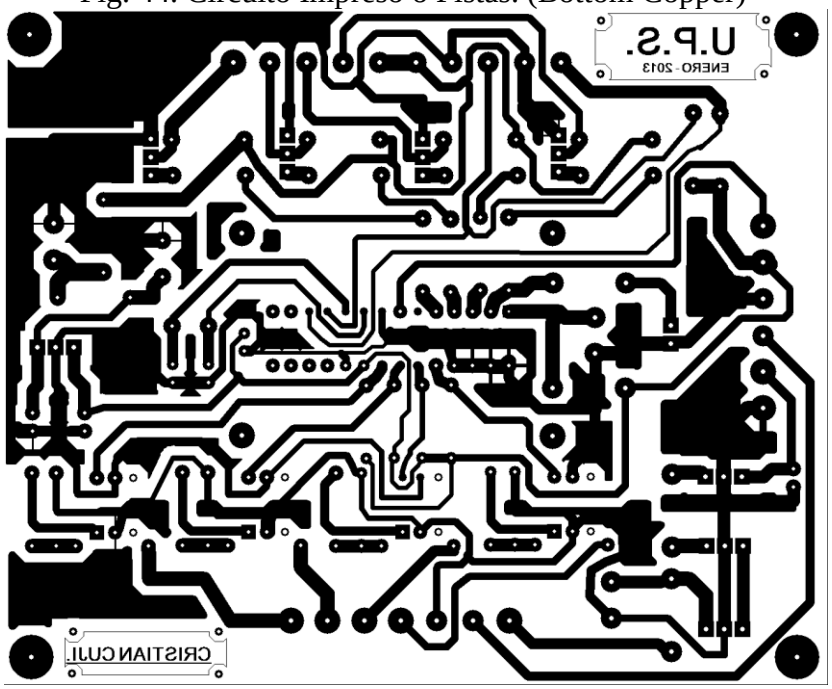
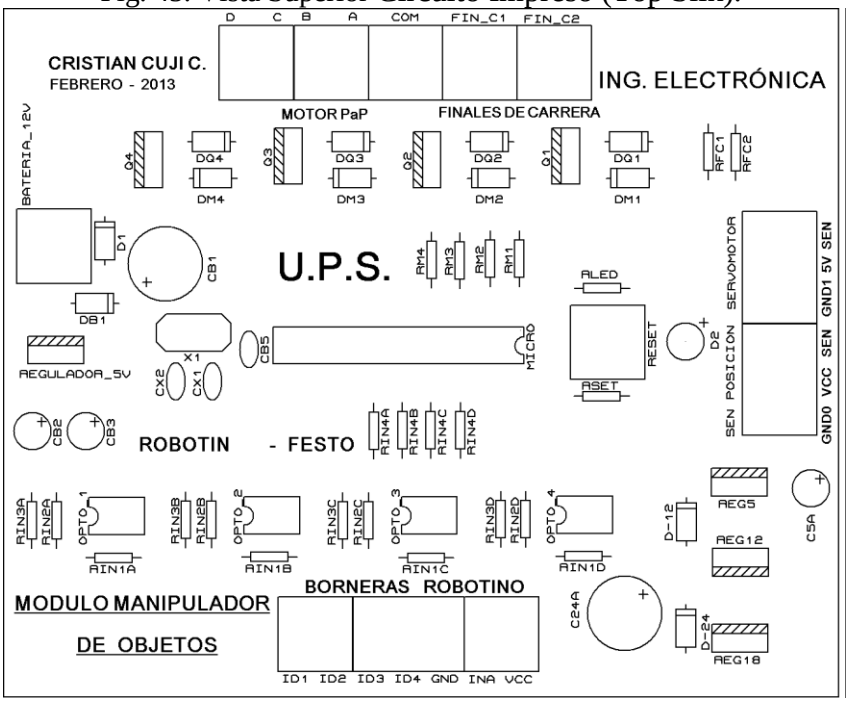
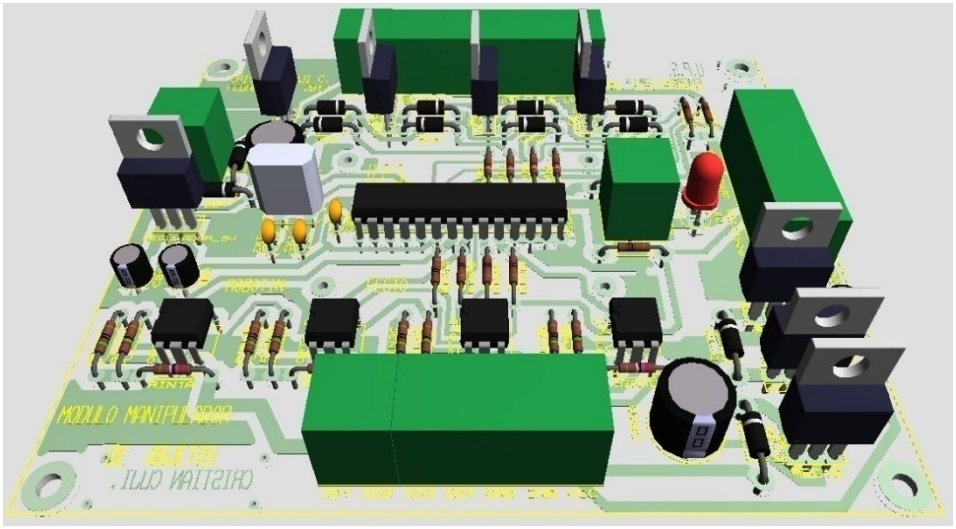


Fig. 45. Vista Superior Circuito Impreso (Top Silk).



Fuente: Proteus - ISIS 7
Elaborado por: Cristian Cuji.

Fig. 46. Simulación Grafico en 3D generado en ARES



Fuente: Proteus - ISIS 7
Elaborado por: Cristian Cuji.

El resultado final se lo puede apreciar en el siguiente grafico.

Fig. 47. Resultado Final, Tarjeta de Control y Adquisición de datos

Elaborado por: Cristian Cuji.

3.2.3. Programación en el microcontrolador

El software seleccionado para el desarrollo del programa del microcontrolador es Pic C, el lenguaje de programación es similar al lenguaje C, además cuenta con librerías especializadas en programación de microcontroladores Microchip, este software fue seleccionado por su amigable interfaz, por su simplicidad y sencillez.

A continuación se especifican detalles necesarios para desarrollar el programa de control del Microcontrolador.

3.2.3.1. Caracterización de sistema

El sistema cuenta con 10 entradas y 5 salidas digitales, con las siguientes especificaciones.

Tabla- 10. Caracterización de entradas / salidas del Microcontrolador

Entradas

No de PIN	Descripción	Bit
Pin 21	Entrada 1	RB0
Pin 22	Subir 1(Activado) 0(Detenido)	RB1
Pin 23	Bajar 1(Activado) 0(Detenido)	RB2
Pin 24	Gripper 1(Abierto) 0(Cerrado)	RB3
Pin 11	Sensor final de carrera ALTO	RC0
Pin 12	Sensor final de carrera BAJO	RC1

Salidas.

No de PIN	Descripción	Bit
Pin 2	Bobina 1 del motor	RA0
Pin 3	Bobina 2 del motor	RA1
Pin 4	Bobina 3 del motor	RA2
Pin 5	Bobina 4 del motor	RA3
Pin 7	Señal de control del servomotor.	RA5

Elaborado por: Cristian Cuji.

3.2.3.2. Descripción del programa

Se requiere identificar 8 entradas digitales provenientes del Robotino y 2 entradas de sensores finales de carrera. Las condiciones necesarias para desarrollar el programa son las siguientes:

- RB0: Es un bit de validación permite cambiar el mando de: 1L (Activación del sistema), 0L (Apagar sistema)
- RB1: Permite subir la pinza haciendo girar el motor en sentido horario, siempre que RB0 se encuentre en 1L.
- RB2: Permite bajar la pinza haciendo girar el motor en sentido anti horario, siempre que RB0 se encuentre en 1L.
- RB3: Es el control del Gripper o Pinza, 1L pinza abierta / 0L pinza cerrada.

Las entradas digitales mencionadas deben controlar las salidas del sistema, principalmente la activación del servomotor, control de pulsos para el movimiento del motor de pasos y la seguridad de los sensores final de carrera:

- RA5: Es el control de pasos del servomotor, esto se consigue generando tren de pulsos (2400us para abrir la pinza) y (1200us para cerrar la pinza).
- RA0 a RA3: Son los puertos que generan los pulsos para el control del motor de pasos.

Tabla- 11. Secuencia de control del motor de pasos

Horario SUBIR				
RA0	**RA1**	**RA2**	**RA3**	**Activa**
1	0	0	0	Bobina A
0	1	0	0	Bobina B
0	0	1	0	Bobina C
0	0	0	1	Bobina D

Anti horario BAJAR				
RA0	**RA1**	**RA2**	**RA3**	**Activa**
0	0	0	1	Bobina D
0	0	1	0	Bobina C
0	1	0	0	Bobina B
1	0	0	0	Bobina A

Apagado				
RA0	**RA1**	**RA2**	**RA3**	**Activa**
0	0	0	0	Motor detenido

Elaborado por: Cristian Cuji.

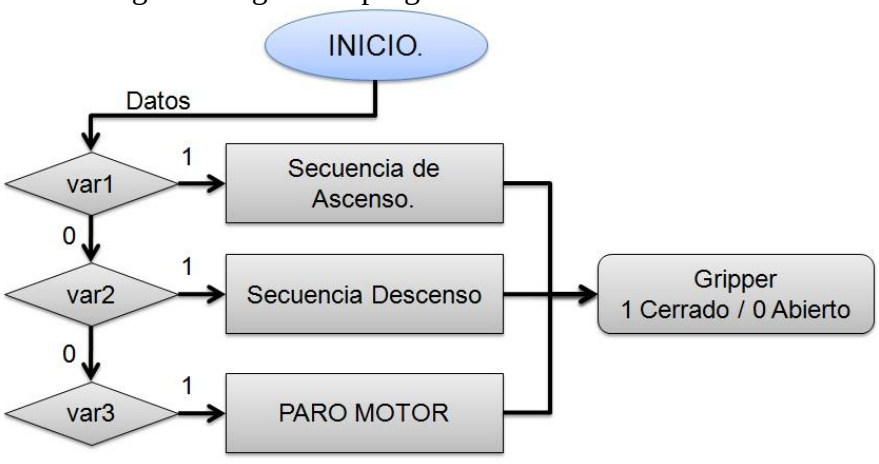
El transistor se activa por la base, cerrando el circuito entre colector y emisor. La bobina y el voltaje de referencia (tierra) se activan por medio del transistor, esto se produce de manera secuencial activando las bobinas en forma en forma ordenada y secuencial como lo indica la Tabla- 11. Secuencia de control del motor de pasos

3.2.3.3. Algoritmo y desarrollo de programa para el microcontrolador

De acuerdo a la descripción de programa el algoritmo y el código es el siguiente:

Algoritmo:

Fig. 48. Algoritmo programación microcontrolador



Elaborado por: Cristian Cuji.

Código Fuente:

Fig. 49.- Código Fuente del Programa en el Microcontrolador

```
#include "C:\Documents and Settings\sistemas\Escritorio
\ProgramaFinalTesisPIC\DEFINITIVO\ProgramaDefinitivo\Movimientos_PIC.h"

// Configuración de puertos de entrada y salida del microcontrolador.

#define    OFF      output_low          // Salida en bajo
#define    ON       output_high         // Salida en alto
#define    B1         PIN_C0
#define    B2         PIN_C1
#define    Q1         PIN_A0
#define    Q2         PIN_A1
#define    Q3         PIN_A2
#define    Q4         PIN_A3
#define    servo       PIN_A5

// velocidad del motor  a pasos 25 ms

// Definición de variables utilizadas en el programa.
int8  dato=0,sube;
int8  dato_1=0;
int16 dato_2=0;
int   primera =0;
int   paso=0;
void  paro();
void  hor();
void  ahor();

void main() {
  int16 l;
  int8 k;
  int16 cnt=0;
  set_tris_c(255); //entrada
  set_tris_b(255); //entrada
  set_tris_a(0);   //salida

  OFF(Q1);OFF(Q2);OFF(Q3);OFF(Q4);
     paso=0;

// Estructura inicial del programa en el microcontrolador.

  do {

//Lectura de datos de entrada en los puertos del microcontrolador.

          dato= input_c();
          dato_1=input_b();
             paso=0;
             cnt=0;
             dato_1=input_b() & 3;

// Sentencia de paro de emergencia.
```

```
            if(dato_1==0 ){paro();}

// Sentencia de lectura de giro en sentido horario.

            if(dato_1==1 ){
              if (bit_test(dato,1)==1)
                 hor();
              else
                 paro();
            }

// Sentencia de lectura de giro en sentido antihorario.

            if(dato_1==2){
              if (bit_test(dato,0)==1)
                 ahor();
              else
                 paro();
            }
            dato_1=input_b() & 4;

// Sentencia de lectura de activación del servomotor.

            if (dato_1==4){
              on(servo);delay_us(8800);
              off(servo);delay_ms(80);
            }
            else
            {
              on(servo);delay_us(2400);
              off(servo);delay_ms(80);
            }
          }
  }while (TRUE);
}

// Estructura del void de secuencia de giro del motor de pasos en sentido horario.

void hor()
{
        ON(Q1);  OFF(Q2); OFF(Q3); OFF(Q4); DELAY_MS(25);
        OFF(Q1); ON(Q2);  OFF(Q3); OFF(Q4); DELAY_MS(25);
        OFF(Q1); OFF(Q2); ON(Q3);  OFF(Q4); DELAY_MS(25);
        OFF(Q1); OFF(Q2); OFF(Q3); ON(Q4);  DELAY_MS(25);
}


// Estructura del void de secuencia de giro del motor de pasos en sentido antihorario.

void Ahor()
{
        OFF(Q1); ON(Q2);  OFF(Q3); OFF(Q4); DELAY_MS(25);
        ON(Q1);  OFF(Q2); OFF(Q3); OFF(Q4); DELAY_MS(25);
        OFF(Q1); OFF(Q2); OFF(Q3); ON(Q4);  DELAY_MS(25);
        OFF(Q1); OFF(Q2); ON(Q3);  OFF(Q4); DELAY_MS(25);
```

```
}

// Estructura del void de secuencia de giro del motor de pasos en paro.

void paro()
{
        OFF(Q1); OFF(Q2); OFF(Q3); OFF(Q4);
}
}
```

Fuente: Software PicC.
Elaborado por: Cristian Cuji.

Fig. 50.- Características Adicionales del Código Fuente

Características Adicionales del Código Fuente:	
Puerto C y Puerto B	Son entradas digitales: Puerto B son 4 entradas digitales provenientes del Robotino. Puerto C son 2 entradas digitales provenientes de los dos sensores fin de carrera colocados por seguridad.
Servomotor.	El Servomotor es de marca HiTech, los datos que necesita para abrirse y cerrarse son los siguientes: on(servo);delay_us(8800); Para abrirse. on(servo);delay_us(2400); Para cerrarse
Motor de pasos.	Se controlan con pulsos de 50 miliSegundos en cada bobina. Entre menor el tiempo de excitación en la bobina menor riesgo de producir efectos de temperatura excesiva en el motor. Y por seguridad y se recomienda apagar todas las bobinas del motor. El tornillo sin fin posee un freno natural, evitando que la pinza regrese o seda en su posición debido al peso.

Elaborado por: Cristian Cuji.

3.3. Implementación en el robotino

Robotino View es un software diseñado por FESTO, exclusivamente para el control del Sistema Robot Móvil Robotino. Para el desarrollo del proyecto se planteo el uso de este software de control por las siguientes razones:

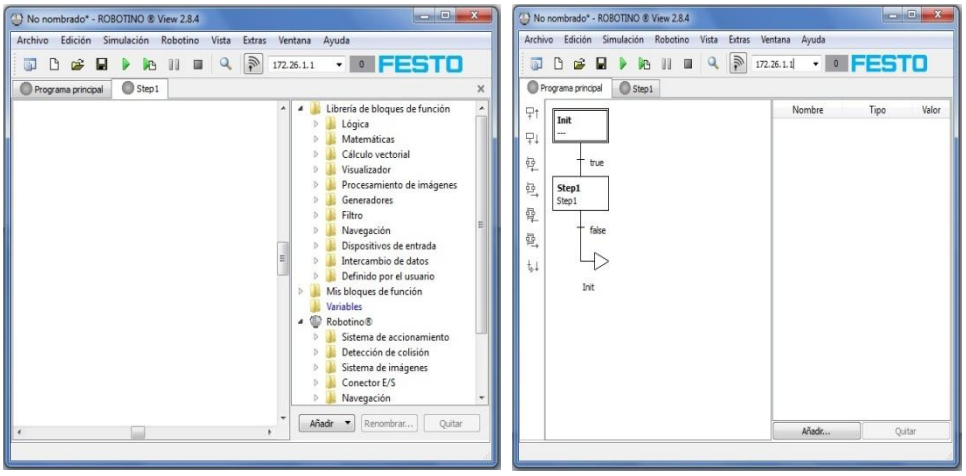
- Crear un driver exclusivo para el sistema manipulador de objetos.
- Es fácilmente adaptable el puerto de entradas / salidas a nuestro proyecto.

- Es posible generar un icono driver por medio de programación en C++.

3.3.1. Robotino View

Robotino View es un software de programación que posee un entorno grafico e intuitivo para crear y ejecutar programas de control para el Robot Móvil Robotino ®. El entorno grafico del software posee dos entornos uno de programación en Gráfica (Step1) y Grafset (Programa principal).

Fig. 51. Ventanas de Programación en Robotino View 2.8.4. (Programación Gráfica y Grafset)



Fuente: Software Robotino View v2.8.4.
Elaborado por: Cristian Cuji.

Sus principales características son:

- Diseño de programas de control de flujo, por medio de librerías de bloques de funciones con las cuales se pueden desarrollar un grafico de flujo de datos. Se conoce más comúnmente como programación Grafset.
- Programación en lenguaje gráfico e intuitivo en varios subprogramas, integrados en un programa general por medio de flujo de datos. Los subprogramas pueden ser importados desde diferentes proyectos.
- Robotino® View 2 no sólo es capaz de controlar un Robotino, sino cualquier dispositivo y en cantidades ilimitadas. Por ejemplo, puede controlarse cualquier número de Robotinos simultáneamente desde un proyecto Robotino View.

- Diseñar e implementar un "bloque de función" personalizado que se hayan cargado en la librería de bloques de función como Plugin en el administrador de dispositivos.
- El diseño de iconos personalizados se lo puede realizar con herramientas propias de Robotino View y API Robotino, y ser editadas en un compilador de lenguaje C++.

3.3.2. API (Application Programming Interface)

API (Application Programming Interface), una traducción aproximada puede ser Aplicación de Interfaz de Programación. El objetivo es tener acceso a librerías propias del Sistema Robot Móvil Robotino, por medio de un lenguaje de programación por ejemplo: C, C++, C#, Java, .NET, LabView, MatLab, entre otros.

RobotinoView_API_v2.8.4 es un software que al ser instalado proporciona herramientas para generar y traducir códigos fuentes que pueden ser editados para controlar el funcionamiento de un bloque de funciones personalizado que va a funcionar como un nuevo icono en RobotinoView_v2.8.4.

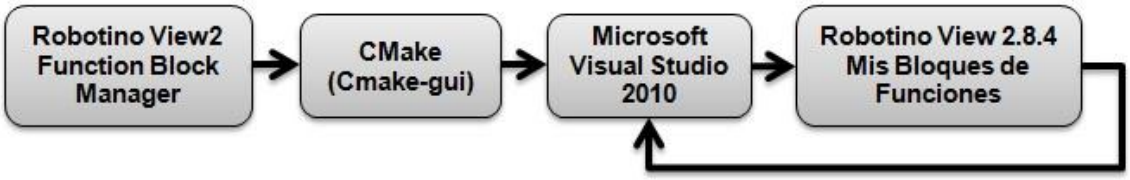
3.3.3. Bloque de funciones

Los detalles de la configuración y las herramientas necesarias que se debe descargar se encuentran en el Anexo 1. HERRAMIENTA "BLOQUE DE FUNCIONES" PARA ROBOTINO VIEW 2.8.4.

Para trabajar con la herramienta MIS BLOQUES DE FUNCIONES, es necesario instalar RobotinoView_v2.8.4, se instala automáticamente: Function Block Manager y CMaker-gui. Adicionalmente es necesario contar con un compilador como Visual Studio 2010.

El proceso de elaboración de un nuevo bloque de funciones es el siguiente:

Fig. 52. Proceso de elaboración del Bloque de Funciones



Elaborado por: Cristian Cuji.

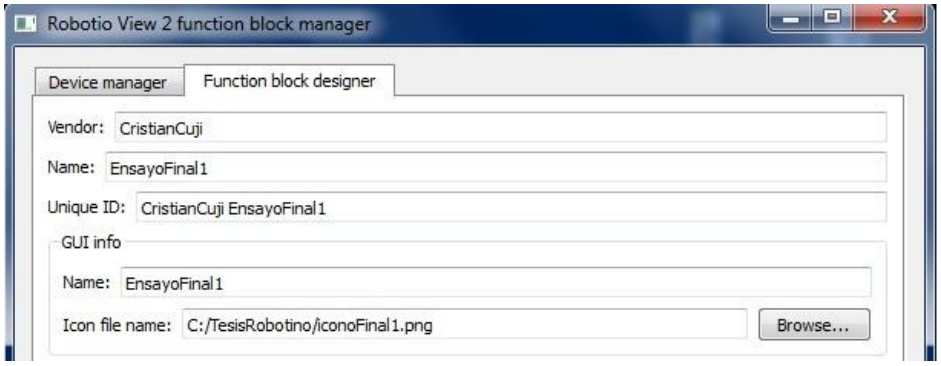
3.3.3.1. Function Block Manager

La herramienta permite crear un nuevo bloque de funciones o dicho de otra forma un nuevo Icono. Es necesario asignar un nombre, un icono en formato (.PNG) y agregar el cantidad y el tipo de entradas, salidas y variables necesarias para desarrollar el programa, este paso no puede ser editado por lo que se debe planificar bien el número de entradas, salidas y también el numero de variables que se van a utilizar en nuevo bloque de funciones. Además la misma herramienta puede permitir al icono ser visible o invisible en el software Robotino View 2.8.4.

Esta herramienta se instala junto con la herramienta API (Application Programming Interface), que puede ser descargada desde la página web de Festo. Se puede visualizar en el Menú Inicio / Programas / Function Block Manager.

Solo es necesario llenar los campos Vendor, Name y Seleccionar Icon file name. Dar clik en AutoFill y los demás campos se llenan automáticamente.

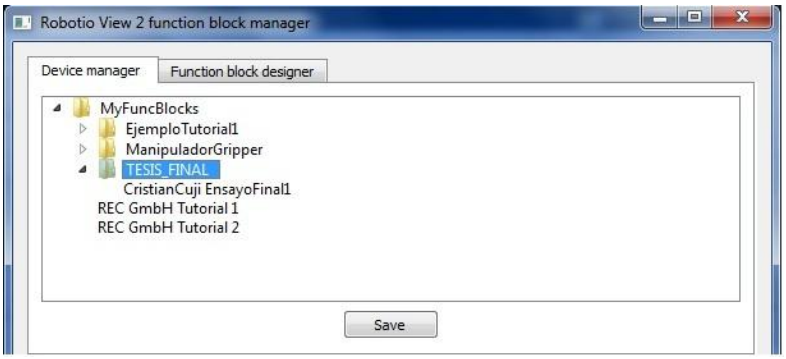
Fig. 53.Function Block Manager - Configuración Inicial



Fuente: Software Function Block Manager.
Elaborado por: Cristian Cuji.

En Device manager se configura la ubicación que va tener dentro del software RobotinoView, se puede crear un nuevo directorio y agregar el nuevo bloque de funciones. Y con SAVE se guardan los cambios, en este punto se puede visualizar desde RobotinoView el nuevo bloque de funciones.

Fig. 54.Function Block Manager - Device Manager



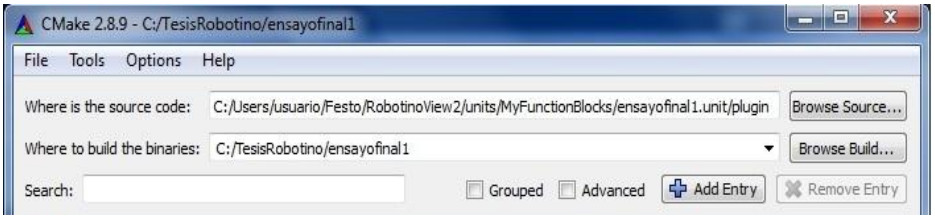
Fuente: Software Function Block Manager.
Elaborado por: Cristian Cuji.

3.3.3.2. CMake

CMake permite traducir un código generado como "plugin" desde Function Block Manager a un código traducido en un compilador previamente seleccionado, para este proyecto se escogió a Visual Studio 2010. Esta herramienta se instala junto con la instalación del API y se visualiza en el menú de Inicio / Programas / CMake.

CMaker necesita saber dos direcciones, la una dirección de fuente (Browse Source) y una dirección de trabajo o edición (Browse Build). La dirección de Source es una dirección creada en el paso anterior por Function Block Manager. Y la dirección de Build es una dirección establecida por el programador, es una dirección personal donde se van a generar y configurar el nuevo proyecto.

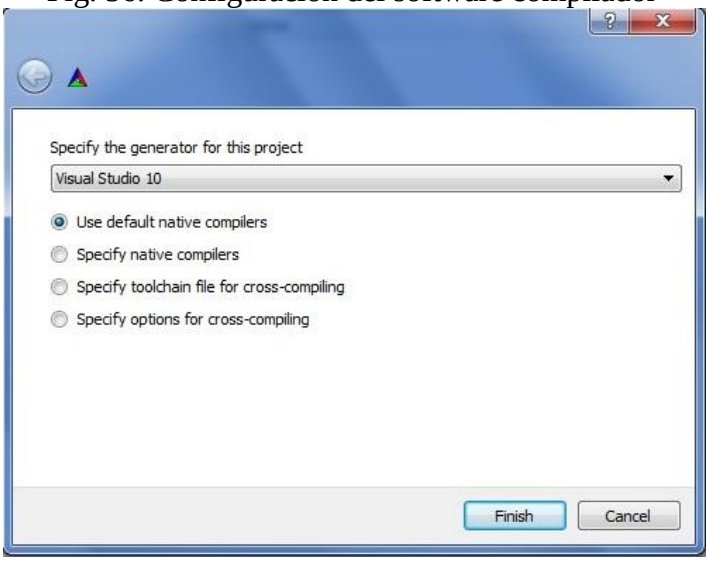
Fig. 55. Configuración de direcciones de CMaker-gui



Fuente: Software CMaker 2.8.9.
Elaborado por: Cristian Cuji.

Con las dos direcciones establecidas se procede a configurar la traducción de código, CMaker - gui pregunta que software compilador se tiene disponible generando una lista de la que podemos escoger por ejemplo: Visual Studio 2010.

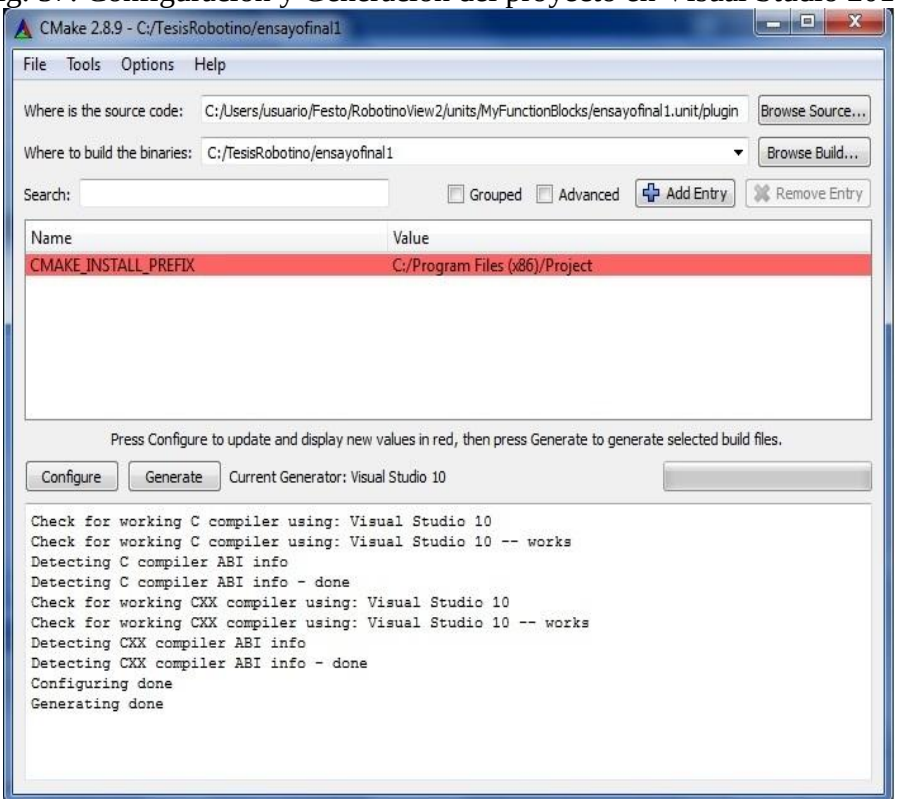
Fig. 56. Configuración del software compilador



Fuente: Software CMaker 2.8.9.
Elaborado por: Cristian Cuji.

Tarda unos segundos en configurarse el nuevo proyecto y luego se procede a configurar el nuevo proyecto.

Fig. 57. Configuración y Generación del proyecto en Visual Studio 2010

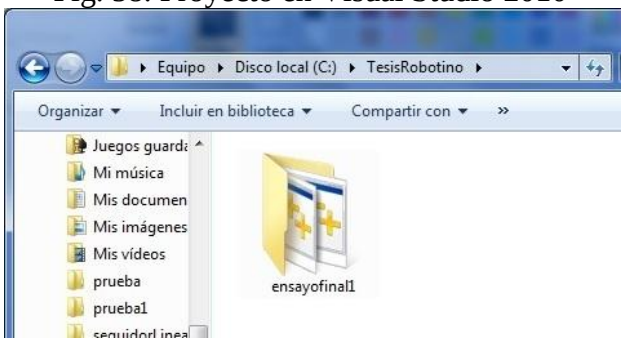


Fuente: Software CMaker 2.8.9.
Elaborado por: Cristian Cuji.

3.3.3.3. Microsoft Visual Studio 2010

Visual Studio 2010 fue el compilador seleccionado por la facilidad de adquisición, disponibilidad e instalación. La programación se lo realiza en lenguaje C++. El texto que se tomo pare guía en la programación fue (JOYANES AGUILAR & ZAHONERO MARTÍNEZ, 2010).

Fig. 58. Proyecto en Visual Studio 2010



Elaborado por: Cristian Cuji.

Es importante considerar que el compilador no debe ejecutar la compilación al programa porque se generan errores producidos por el API. Lo correcto es Generar la Solucion (F6) del programa. De esta manera el software no se fija en los errores del API y generar el código de manera correcta para ser traducida y entendida desde Robotino View.

3.3.3.4. Nuevo bloque de funciones

El nuevo bloque de funciones es un icono que puede ser visualizado en Robotino View 2.8.4. Dentro de la carpeta “Mis bloques de funciones”.

El icono contiene las instrucciones previamente programadas en Visual Studio 2010, posee el número de entradas, salidas y el gráfico seleccionado en Function Block Manager. Este icono es utilizado en el entorno de programación de Robotino View y ser editado solo desde Visual Studio, cualquier cambio en la estructura física como puede ser aumentar o quitar una entrada o una salida no se puede editar, necesariamente se requiere desarrollar un nuevo bloque de funciones.

3.3.4. Implementación de driver de control por medio de un bloque de funciones

Para mayor información de descarga de software, versiones, instalación y configuración se recomienda revisar el anexo ““BLOQUE DE FUNCIONES” PARA ROBOTINO VIEW 2.8.4.”.

El presente apartado es una recopilación del desarrollo del driver de control desarrollado en el proyecto de tesis.

3.3.4.1. Funcionamiento del driver del control

El driver es un icono que facilita la programación del módulo manipulador de objetos, consta de 4 entradas y 5 salidas, que a continuación se detallan.

Tabla- 12. Descripción de entradas del icono de control

Entradas				
Nombre	**Tipo de Variable**	**Conector**		
ON/OFF	Bool	Interno		
Posición	Int	Interno		
Pinza	Bool	Entrada Digital 6	DIN6	Pin 4
Sensor	Float	Entrada Analógica 1	IAN1	Pin14

Elaborado por: Cristian Cuji.

- ON/OFF: Enciende o apaga la tarjeta de control.
- Posición: Es un valor del 0 al 10 que indica la posición en la que se desea ubicar al gripper.
- Pinza: 0 Abrir pinza / 1 Cerrar pinza, siempre que la tarjeta se encuentre encendida.
- Sensor: es el valor que se recibe desde el sensor de posición.

Tabla- 13. Descripción de salidas del Icono de control

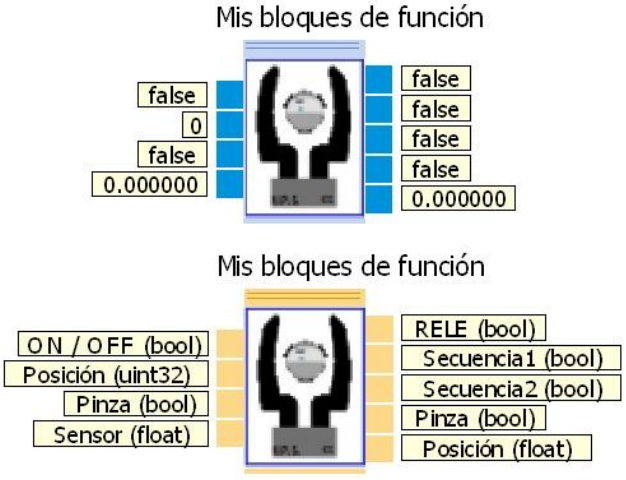
Salidas				
Nombre	**Tipo de Variable**	**Conector**		
Rele	Bool	Relé 1	Ain 1	Pin19 - Pin20
Secuencia1	Bool	Salida digital 8	DO8	Pin6
Secuencia2	Bool	Salida digital 7	DO7	Pin7
Pinza	Bool	Salida digital 6	DO6	Pin8
Posición	Int	Salida digital 5	DO5	Pin9

Elaborado por: Cristian Cuji.

- Relé: Se activa o desactiva como un switch para encender o apagar la tarjeta de control.
- Secuencia 1: Inicia la secuencia de giro en sentido horario es decir permite ascender al gripper.
- Secuencia 2: Inicia la secuencia de giro en sentido anti horario es decir permite descender al gripper.
- Pinza: Tiene el control de apertura o cierre del gripper o pinza.
- Posición: Indica la posición en la que se encuentra el gripper.

El icono que se visualiza en Robotino View 2.8.4 es el siguiente en dos posibles visualizaciones: descripción de conector y valores de conector.

Fig. 59. Driver de control, descripción de conectores y valor de conectores



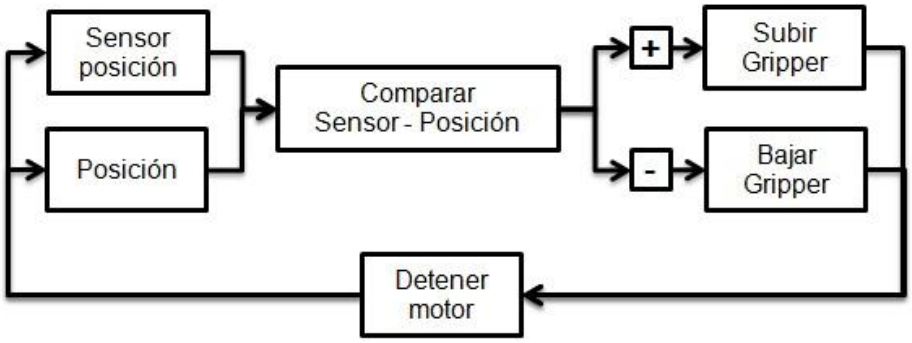
Elaborado por: Cristian Cuji.

3.3.4.2. Descripción del algoritmo

El sensor lee la posición del gripper y compara con la posición deseada, en el interior del icono se realiza una resta entre el valor del sensor y posición (Var1 = Sensor - Posición). El sistema electrónico ascenderá o descenderá según sea necesario hasta que la comparación sea cero.

Con ON/OFF se puede activar la tarjeta electrónica y solo en ese momento se puede abrir o cerrar el gripper.

Fig. 60. Algoritmo de programación de ascenso y descenso del gripper



Elaborado por: Cristian Cuji.

Al modulo manipulador de objetos se le ha dividido en 12 posibles posiciones, en un rango de libertad de 15cm. La tabla siguiente muestra la distribución de las posiciones y su equivalencia en centímetros.

Tabla- 14. Ubicación y descripción de cada posición

Posición	Ubicación	Descripción
-1	0,1 cm	Punto final Inferior
0	0,5 cm	Capturar del piso 1
1	1 cm	Capturar del piso 2
2	3,5 cm	Posición elevada 1
3	5 cm	Posición elevada 2
4	6,5 cm	Posición elevada 3
5	8 cm	Posición elevada 4

6	9,5 cm	Posición elevada 5
7	11 cm	Posición elevada 6
8	12,5 cm	Posición elevada 7
9	14 cm	Posición elevada 8
10	14,5 cm	Punto final Superior

Elaborado por: Cristian Cuji.

La posición -1 es una posición con la que prácticamente se raspa el piso con el gripper, por esta razón es recomendable utilizar dos posiciones de aproximación al suelo, la posición 1 y 2. Desde la posición 3 al 9 son posiciones elevadas y la posición 10 es una posición de seguridad en el punto final superior.

Esta descripción de programación es transparente gracias a Cmake - gui y a la herramienta Function Block Manager.

3.3.4.3. Algoritmo de Programación en Visual Studio 2010

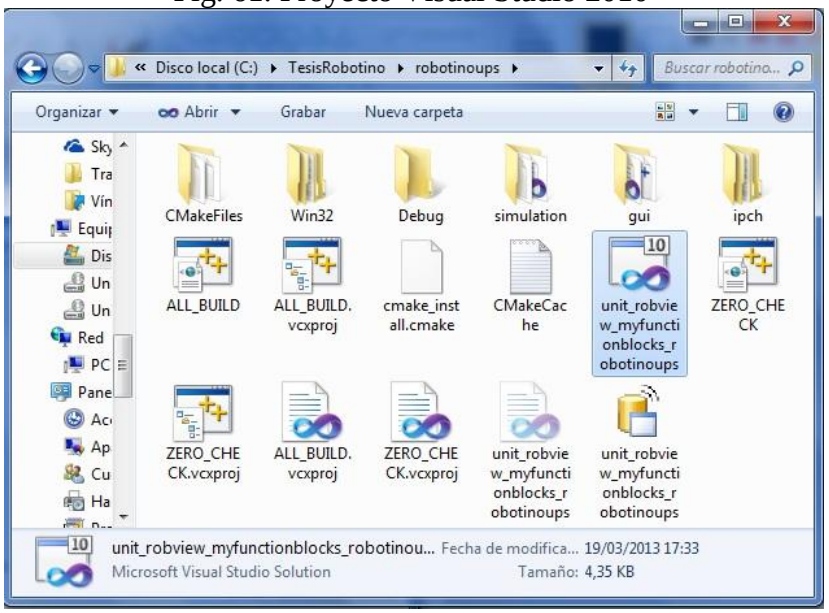
Una vez generado el código desde la herramienta Function Block Manager y traducido por CMake - gui, se obtiene una carpeta que contiene el código fuente en Visual Studio 2010 que fue mi compilador seleccionado. El nombre del proyecto generalmente es similar: "*unit_robview_myfunctionblocks_ensayofinal1"*

Una vez abierto el proyecto se despliega la ventana de Visual Studio 2010, el proyecto posee dos subproyectos:

- *unit_robview_myfunctionblocks_ensayofinal1_gui*
- *unit_robview_myfunctionblocks_ensayofinal1_simulation*

Y ambos proyectos con un archivo en particular pero diferente en cada uno "EnsayoFinal1.cpp"

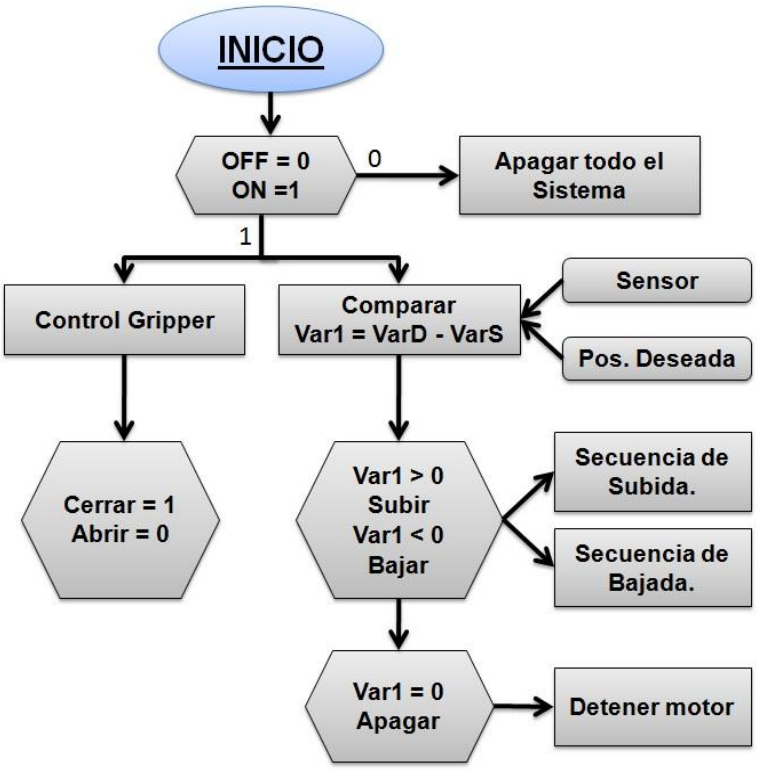
Fig. 61. Proyecto Visual Studio 2010



Elaborado por: Cristian Cuji.

En el presente subcapítulo presenta la explicación del funcionamiento y comandos utilizados para el desarrollo del proyecto.

Fig. 62. Algoritmo del Driver de Control



Elaborado por: Cristian Cuji.

El Driver funciona principalmente con una señal de encendido y apagado. El apagado, Off o cero lógico. Bloquea y deja sin funcionamiento todo el sistema apagándolo directamente de la batería.
Colocando en ON o 1 lógico el Sistema está listo para trabajar con dos rutinas: Control del Gripper y un Comparador. El control del Gripper permite abrir y cerrar la pinza para sujetar o soltar los objetos.

El comparador toma dos valores: Sensor y Posición. La posición (VarD) es un valor deseado al que se quiere alcanzar. El sensor (VarS) es un dispositivo análogo que permite conocer la posición del gripper. La resta de VarD - VarS presenta como resultado una variable nueva (Var1) que puede ser un valor positivo o negativo según los datos, de acuerdo al valor de Var1 el sistema activa la secuencia de subida o bajada. Cuando los valores de Sensor y Posición son iguales Var1 es igual a cero “0”, produciendo que el motor se detenga por el motivo de que alcanzo el valor o la posición esperada.

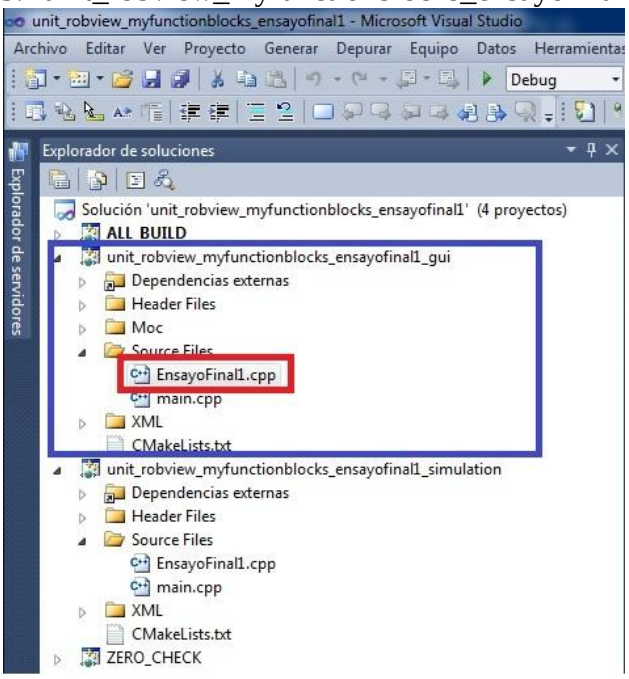
Principales comandos
del.registerInput("in0_Act", 1); Es un código generado por defecto en la traducción del CMake desde el codigo de Manager Function Block para variables de entrada, es importarte que no esté comentado para evitar errores al momento de generar la solución
del.registerOutput("out0_Rel", 1); Código generado por defecto en la traducción del CMaker para variables de salida.
del.registerStateVariable("var0f", 1); Código generado por defecto en la traducción del CMake para variables.
in0_Act = readInput("in0_Act").toBool(); Es la lectura de la variable de entrada, esta línea es obligatoria para lectura de las entradas al Bloque personalizado.
out0_Rel = in0_Act; Se debe asignar un valor o variable al puerto de salida.
writeOutput("out0_Rel", out0_Rel); Escribe el valor asignado en la variable de salida.

Estas palabras reservadas o comandos se repiten constantemente en todo el código fuente, para mayores informaciones un libro o tutoriales de Programación en C++ puede brindar mayores detalles.

3.3.5. Desarrollo de Programación del Driver de control

3.3.5.1. UNIT_ROBVIEW_MYFUNCTIONBLOCKS_ENSAYOFINAL1_GUI

Fig. 63.- unit_robview_myfunctionblocks_ensayofinal1_gui



Fuente: Microsoft Visual Studio 2010.
Elaborador por: Cristian Cuji.

La principal característica en este subproyecto es evitar que alguno de las líneas de cogido que declaran las variables, entradas y salidas no estén comentadas. Por defecto las líneas de código se encuentran comentadas. Es necesario para evitar errores des comentar las variables que se van a utilizar o mejor des comentar todas las líneas.

Linea comentada

```
//del.registerInput("in0_Act", 1);
```

Linea sin comentario.

```
del.registerInput("in0_Act", 1);
```

Aparte de este procedimiento no se recomienda hacer nada más. Aunque con mayores conocimientos en programación se podrían hacer mayores aplicaciones.

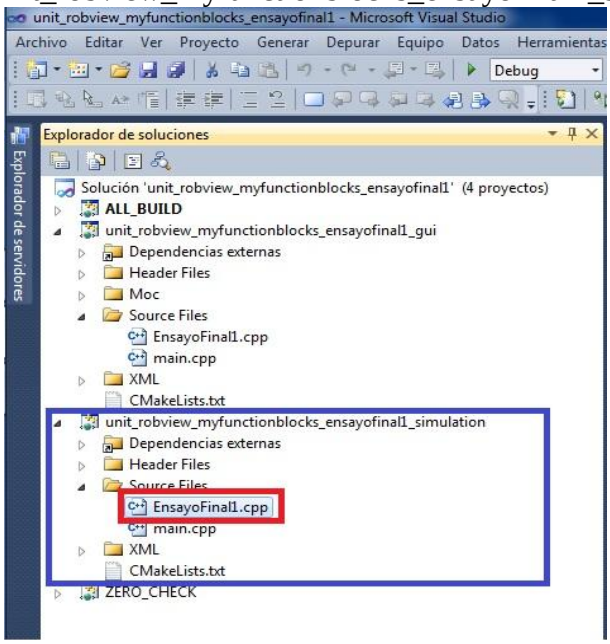
Fig. 64.- Código fuente "unit_robview_myfunctionblocks_ensayofinal1_gui"

```
#include "EnsayoFinal1.h"

EnsayoFinal1::EnsayoFinal1 (plugin::gui::UnitDelegate& del)
: plugin::gui::UnitDialog(del)
{
/*Todo el código es generado automáticamente por el Software CMaker, cuando se generar y configuran las entradas, salidas y variables. Para el funcionamiento es necesario descomentar todos los registros.*/
        // If the dialog is supposed to be notified when a input's, output's or state
           variable's value changes,
        // just uncomment the corresponding line below and modify the update() method.

/*Registros de entrada*/

        del.registerInput("in0_Act", 1);
        del.registerInput("in1_Pos", 1);
        del.registerInput("in2_Gri", 1);
        del.registerInput("in3_Sen", 1);

/*Registros de salida*/

        del.registerOutput("out0_Rel", 1);
        del.registerOutput("out1_Sub", 1);
        del.registerOutput("out2_Baj", 1);
        del.registerOutput("out3_Gri", 1);
        del.registerOutput("out4_Pos", 1);

/*Registros de variables*/

        del.registerStateVariable("var0f", 1);
        del.registerStateVariable("var1f", 1);
        del.registerStateVariable("var2f", 1);
        del.registerStateVariable("var3f", 1);
        del.registerStateVariable("var4f", 1);
        del.registerStateVariable("var5f", 1);
        del.registerStateVariable("var6i", 1);
        del.registerStateVariable("var7i", 1);
        del.registerStateVariable("var8i", 1);
        del.registerStateVariable("var9i", 1);
        del.registerStateVariable("var10u", 1);
        del.registerStateVariable("var11u", 1);
        del.registerStateVariable("var12u", 1);
        del.registerStateVariable("var13u", 1);
        del.registerStateVariable("var14b", 1);
        del.registerStateVariable("var15b", 1);
        del.registerStateVariable("var16b", 1);
        del.registerStateVariable("var17b", 1);
}
```

Fuente: Microsoft Visual Studio 2010.
Elaborador por: Cristian Cuji.

3.3.5.2. UNIT_ROBVIEW_MYFUNCTIONBLOCKS_ENSAYOFINAL1_SIMULATION

Fig. 65.- unit_robview_myfunctionblocks_ensayofinal1_simulation



Fuente: Microsoft Visual Studio 2010.
Elaborador por: Cristian Cuji.

La mayor cantidad de lógica desarrollada para el proyecto se encuentra en este subproyecto, Básicamente se declararon variables, locales, entradas / salidas y su respectivo tipo de variable. Y a partir del clase "RobotinoUPS::step" se procedió a desarrollar toda la lógica de programación

Tabla- 15.- Código fuente "unit_robview_myfunctionblocks_ensayofinal1_simulation"

```
#include "RobotinoUPS.h"

/*Configuración inicial, declaración de variables: entradas, salidas y variables*/

int in1_Pos ;
float out4_Pos, in3_Sen;
bool in0_Act, in2_Gri, out0_Rel, out1_Sub, out2_Baj, out3_Gri;
float var0f, var1f, var2f, var3f, var4f, var5f;
int var6i, var7i, var8i, var9u, var10u;
bool var11b, var12b;

/*Código generado automáticamente, es la dirección del archivo generado*/

RobotinoUPS::RobotinoUPS (plugin::simulation::UnitDelegate& del)
: plugin::simulation::Unit(del, plugin::simulation::Deterministic)
{
}
/*Inicio del código fuente de programación, RobotinoUPS es el nombre con el cual se denomino el proyecto en la herramienta Function Block Manager*/
```

```
void RobotinoUPS::step (void)
{
```

/*Lectura de las variables físicas de entrada en el driver, se observa que son variables de distinto tipo: dos variables booleanas, una variable de dato entero y una de valor flotante*/

```
        in0_Act = readInput("in0_Act").toBool();
        in1_Pos = readInput( "in1_Pos" ).toInt();
        in2_Gri = readInput("in2_Gri").toBool();
        in3_Sen = readInput("in3_Sen").toFloat();
```

/*Valores en las salidas físicas del driver, específicamente la activación de la tarjeta electrónica por medio del Relé 1 */

```
        out0_Rel = in0_Act;
        out3_Gri = in2_Gri;
        var0f = in3_Sen*100;
```

/*Inicio de condiciones programadas en el driver de control: `(in0_Act==1)` Indica la activación de la tarjeta electrónica de control*/

/*Condiciones de posicionamiento para la posición # 0*/

```
        if((in1_Pos == 0)&&(in0_Act==1))
        {
```

/*Algoritmo matemático de comparación para posicionar el gripper en la posición requerida*/

```
                var5f = 46,875;
                var2f = var5f - var0f;
                out4_Pos = var2f;
```

/*(var2f) es una variable que entrega un valor flotante positivo o negativo*/

/*Si el valor de (var2f) es mayor a 1 o un valor positivo se activa la variable de subida hasta que el valor (var2f) se encuentre entre -1 y +1. Cuando la variable se encuentre dentro del rango el gripper se encuentra posicionado. Así funciona para todas las posiciones programadas.*/

/*Condición de subida o Ascenso*/

```
                if(var2f>1)
                {
                        out1_Sub=1;
                        out0_Rel=1;
                        writeOutput( "out1_Sub", out1_Sub );
                        writeOutput( "out0_Rel", out0_Rel );
                }else
                {
                        out1_Sub=0;
                        writeOutput( "out1_Sub", out1_Sub );
                }
```

/*Condición de bajada o Descenso*/

```
                if(var2f<-1)
                {
                        out2_Baj=1;    ;
                        out0_Rel=1;
                        writeOutput( "out2_Baj", out2_Baj );
                        writeOutput( "out0_Rel", out0_Rel );
```

```
		}else
		{
			out2_Baj=0;
			writeOutput( "out2_Baj", out2_Baj );
		}
```

/*Condición de Activación (Apertura y Cierre del Gripper)*/

```
		if(var2f<1 && var2f>-1)
		{
			out0_Rel = 0;
			writeOutput( "out0_Rel", out0_Rel );
			out0_Rel = in0_Act;
			out3_Gri = in2_Gri;
			writeOutput( "out0_Rel", out0_Rel );
			writeOutput( "out3_Gri", out3_Gri );
		}

		writeOutput( "out4_Pos", out4_Pos );
	};
```

/*Condiciones de posicionamiento para la posición # 1*/

```
	if((in1_Pos == 1)&&(in0_Act==1))
	{
		var5f = 50,7812;
		var2f = var5f - var0f;
		out4_Pos = var2f;
```

/*Condición de subida o Ascenso*/

```
		if(var2f>1)
		{
			out1_Sub=1;
			out0_Rel=1;
			writeOutput( "out1_Sub", out1_Sub );
			writeOutput( "out0_Rel", out0_Rel );
		}else
		{
			out1_Sub=0;
			writeOutput( "out1_Sub", out1_Sub );
		}
```

/*Condición de bajada o Descenso*/

```
		if(var2f<-1)
		{
			out2_Baj=1;    ;
			out0_Rel=1;
			writeOutput( "out2_Baj", out2_Baj );
			writeOutput( "out0_Rel", out0_Rel );
		}else
		{
			out2_Baj=0;
			writeOutput( "out2_Baj", out2_Baj );
		}
```

/*Condición de Activación (Apertura y Cierre del Gripper)*/

```
		if(var2f<1 && var2f>-1)
		{
```

```
                        out0_Rel = 0;
                        writeOutput( "out0_Rel", out0_Rel );
                        out0_Rel = in0_Act;
                        out3_Gri = in2_Gri;
                        writeOutput( "out0_Rel", out0_Rel );
                        writeOutput( "out3_Gri", out3_Gri );
                }

                writeOutput( "out4_Pos", out4_Pos );
        };
```

/*Condiciones de posicionamiento para la posición # 2*/

```
        if((in1_Pos == 2)&&(in0_Act==1))
        {
                var5f = 58,5937;
                var2f = var5f - var0f;
                out4_Pos = var2f;
```

/*Condición de subida o Ascenso*/

```
                if(var2f>1)
                {
                        out1_Sub=1;
                        out0_Rel=1;
                        writeOutput( "out1_Sub", out1_Sub );
                        writeOutput( "out0_Rel", out0_Rel );
                }else
                {
                        out1_Sub=0;
                        writeOutput( "out1_Sub", out1_Sub );
                }
```

/*Condición de bajada o Descenso*/

```
                if(var2f<-1)
                {
                        out2_Baj=1;    ;
                        out0_Rel=1;
                        writeOutput( "out2_Baj", out2_Baj );
                        writeOutput( "out0_Rel", out0_Rel );
                }else
                {
                        out2_Baj=0;
                        writeOutput( "out2_Baj", out2_Baj );
                }
```

/*Condición de Activación (Apertura y Cierre del Gripper)*/

```
                if(var2f<1 && var2f>-1)
                {
                        out0_Rel = 0;
                        writeOutput( "out0_Rel", out0_Rel );
                        out0_Rel = in0_Act;
                        out3_Gri = in2_Gri;
                        writeOutput( "out0_Rel", out0_Rel );
                        writeOutput( "out3_Gri", out3_Gri );
                }

                writeOutput( "out4_Pos", out4_Pos );
        };
```

/*Condiciones de posicionamiento para la posición # 3*/

```
	if((in1_Pos == 3)&&(in0_Act==1))
	{
		var5f = 70,3125;
		var2f = var5f - var0f;
		out4_Pos = var2f;
```

/*Condición de subida o Ascenso*/

```
		if(var2f>1)
		{
			out1_Sub=1;
			out0_Rel=1;
			writeOutput( "out1_Sub", out1_Sub );
			writeOutput( "out0_Rel", out0_Rel );
		}else
		{
			out1_Sub=0;
			writeOutput( "out1_Sub", out1_Sub );
		}
```

/*Condición de bajada o Descenso*/

```
		if(var2f<-1)
		{
			out2_Baj=1;   ;
			out0_Rel=1;
			writeOutput( "out2_Baj", out2_Baj );
			writeOutput( "out0_Rel", out0_Rel );
		}else
		{
			out2_Baj=0;
			writeOutput( "out2_Baj", out2_Baj );
		}
```

/*Condición de Activación (Apertura y Cierre del Gripper)*/

```
		if(var2f<1 && var2f>-1)
		{
			out0_Rel = 0;
			writeOutput( "out0_Rel", out0_Rel );
			out0_Rel = in0_Act;
			out3_Gri = in2_Gri;
			writeOutput( "out0_Rel", out0_Rel );
			writeOutput( "out3_Gri", out3_Gri );
		}

		writeOutput( "out4_Pos", out4_Pos );
	};
```

/*Condiciones de posicionamiento para la posición # 4*/

```
	if((in1_Pos == 4)&&(in0_Act==1))
	{
		var5f = 78,125;
		var2f = var5f - var0f;
		out4_Pos = var2f;
```

/*Condición de subida o Ascenso*/

```
		if(var2f>1)
		{
			out1_Sub=1;
			out0_Rel=1;
			writeOutput( "out1_Sub", out1_Sub );
			writeOutput( "out0_Rel", out0_Rel );
		}else
		{
			out1_Sub=0;
			writeOutput( "out1_Sub", out1_Sub );
		}
```

/*Condición de bajada o Descenso*/

```
		if(var2f<-1)
		{
			out2_Baj=1;   ;
			out0_Rel=1;
			writeOutput( "out2_Baj", out2_Baj );
			writeOutput( "out0_Rel", out0_Rel );
		}else
		{
			out2_Baj=0;
			writeOutput( "out2_Baj", out2_Baj );
		}
```

/*Condición de Activación (Apertura y Cierre del Gripper)*/

```
		if(var2f<1 && var2f>-1)
		{
			out0_Rel = 0;
			writeOutput( "out0_Rel", out0_Rel );
			out0_Rel = in0_Act;
			out3_Gri = in2_Gri;
			writeOutput( "out0_Rel", out0_Rel );
			writeOutput( "out3_Gri", out3_Gri );
		}

		writeOutput( "out4_Pos", out4_Pos );
	};
```

/*Condiciones de posicionamiento para la posición # 5*/

```
	if((in1_Pos == 5)&&(in0_Act==1))
	{
		var5f = 89,8438;
		var2f = var5f - var0f;
		out4_Pos = var2f;
```

/*Condición de subida o Ascenso*/

```
		if(var2f>1)
		{
			out1_Sub=1;
			out0_Rel=1;
			writeOutput( "out1_Sub", out1_Sub );
			writeOutput( "out0_Rel", out0_Rel );
		}else
		{
			out1_Sub=0;
```

```
                        writeOutput( "out1_Sub", out1_Sub );
                }

/*Condición de bajada o Descenso*/

                if(var2f<-1)
                {
                        out2_Baj=1;    ;
                        out0_Rel=1;
                        writeOutput( "out2_Baj", out2_Baj );
                        writeOutput( "out0_Rel", out0_Rel );
                }else
                {
                        out2_Baj=0;
                        writeOutput( "out2_Baj", out2_Baj );
                }

/*Condición de Activación (Apertura y Cierre del Gripper)*/

                if(var2f<1 && var2f>-1)
                {
                        out0_Rel = 0;
                        writeOutput( "out0_Rel", out0_Rel );
                        out0_Rel = in0_Act;
                        out3_Gri = in2_Gri;
                        writeOutput( "out0_Rel", out0_Rel );
                        writeOutput( "out3_Gri", out3_Gri );
                }

                writeOutput( "out4_Pos", out4_Pos );
        };

/*Condiciones de posicionamiento para la posición # 6*/

        if((in1_Pos == 6)&&(in0_Act==1))
        {
                var5f = 101,5625;
                var2f = var5f - var0f;
                out4_Pos = var2f;

/*Condición de subida o Ascenso*/

        if(var2f>1)
                {
                        out1_Sub=1;
                        out0_Rel=1;
                        writeOutput( "out1_Sub", out1_Sub );
                        writeOutput( "out0_Rel", out0_Rel );
                }else
                {
                        out1_Sub=0;
                        writeOutput( "out1_Sub", out1_Sub );
                }

/*Condición de bajada o Descenso*/

                if(var2f<-1)
                {
                        out2_Baj=1;    ;
                        out0_Rel=1;
                        writeOutput( "out2_Baj", out2_Baj );
```

```
                        writeOutput( "out0_Rel", out0_Rel );
                }else
                {
                        out2_Baj=0;
                        writeOutput( "out2_Baj", out2_Baj );
                }
```

/*Condición de Activación (Apertura y Cierre del Gripper)*/

```
                if(var2f<1 && var2f>-1)
                {
                        out0_Rel = 0;
                        writeOutput( "out0_Rel", out0_Rel );
                        out0_Rel = in0_Act;
                        out3_Gri = in2_Gri;
                        writeOutput( "out0_Rel", out0_Rel );
                        writeOutput( "out3_Gri", out3_Gri );
                }

                writeOutput( "out4_Pos", out4_Pos );
        };
```

/*Condiciones de posicionamiento para la posición # 7*/

```
        if((in1_Pos == 7)&&(in0_Act==1))
        {
                var5f = 121,077;
                var2f = var5f - var0f;
                out4_Pos = var2f;
```

/*Condición de subida o Ascenso*/

```
                if(var2f>1)
                {
                        out1_Sub=1;
                        out0_Rel=1;
                        writeOutput( "out1_Sub", out1_Sub );
                        writeOutput( "out0_Rel", out0_Rel );
                }else
                {
                        out1_Sub=0;
                        writeOutput( "out1_Sub", out1_Sub );
                }
```

/*Condición de bajada o Descenso*/

```
                if(var2f<-1)
                {
                        out2_Baj=1;    ;
                        out0_Rel=1;
                        writeOutput( "out2_Baj", out2_Baj );
                        writeOutput( "out0_Rel", out0_Rel );
                }else
                {
                        out2_Baj=0;
                        writeOutput( "out2_Baj", out2_Baj );
                }
```

/*Condición de Activación (Apertura y Cierre del Gripper)*/

```
                if(var2f<1 && var2f>-1)
```

```
		{
			out0_Rel = 0;
			writeOutput( "out0_Rel", out0_Rel );
			out0_Rel = in0_Act;
			out3_Gri = in2_Gri;
			writeOutput( "out0_Rel", out0_Rel );
			writeOutput( "out3_Gri", out3_Gri );
		}

		writeOutput( "out4_Pos", out4_Pos );
	};
```

/*Condiciones de posicionamiento para la posición # 8*/

```
	if((in1_Pos == 8)&&(in0_Act==1))
	{
		var5f = 144,5312;
		var2f = var5f - var0f;
		out4_Pos = var2f;
```

/*Condición de subida o Ascenso*/

```
		if(var2f>1)
		{
			out1_Sub=1;
			out0_Rel=1;
			writeOutput( "out1_Sub", out1_Sub );
			writeOutput( "out0_Rel", out0_Rel );
		}else
		{
			out1_Sub=0;
			writeOutput( "out1_Sub", out1_Sub );
		}
```

/*Condición de bajada o Descenso*/

```
		if(var2f<-1)
		{
			out2_Baj=1;   ;
			out0_Rel=1;
			writeOutput( "out2_Baj", out2_Baj );
			writeOutput( "out0_Rel", out0_Rel );
		}else
		{
			out2_Baj=0;
			writeOutput( "out2_Baj", out2_Baj );
		}
```

/*Condición de Activación (Apertura y Cierre del Gripper)*/

```
		if(var2f<1 && var2f>-1)
		{
			out0_Rel = 0;
			writeOutput( "out0_Rel", out0_Rel );
			out0_Rel = in0_Act;
			out3_Gri = in2_Gri;
			writeOutput( "out0_Rel", out0_Rel );
			writeOutput( "out3_Gri", out3_Gri );
		}

		writeOutput( "out4_Pos", out4_Pos );
```

```
        };
```

/*Condiciones de posicionamiento para la posición # 9*/

```
        if((in1_Pos == 9)&&(in0_Act==1))
        {
                var5f = 187,5;
                var2f = var5f - var0f;
                out4_Pos = var2f;
```

/*Condición de subida o Ascenso*/

```
                if(var2f>1)
                {
                        out1_Sub=1;
                        out0_Rel=1;
                        writeOutput( "out1_Sub", out1_Sub );
                        writeOutput( "out0_Rel", out0_Rel );
                }else
                {
                        out1_Sub=0;
                        writeOutput( "out1_Sub", out1_Sub );
                }
```

/*Condición de bajada o Descenso*/

```
                if(var2f<-1)
                {
                        out2_Baj=1;   ;
                        out0_Rel=1;
                        writeOutput( "out2_Baj", out2_Baj );
                        writeOutput( "out0_Rel", out0_Rel );
                }else
                {
                        out2_Baj=0;
                        writeOutput( "out2_Baj", out2_Baj );
                }

                if(var2f<1 && var2f>-1)
                {
                        out0_Rel = 0;
                        writeOutput( "out0_Rel", out0_Rel );
                        out0_Rel = in0_Act;
                        out3_Gri = in2_Gri;
                        writeOutput( "out0_Rel", out0_Rel );
                        writeOutput( "out3_Gri", out3_Gri );
                }

                writeOutput( "out4_Pos", out4_Pos );
        };
```

/*Condiciones de posicionamiento para la posición # 10*/

```
        if((in1_Pos == 10)&&(in0_Act==1))
        {
                var5f = 210,8;
                var2f = var5f - var0f;
                out4_Pos = var2f;
```

/*Condición de subida o Ascenso*/

```
                if(var2f>1)
                {
                        out1_Sub=1;
                        out0_Rel=1;
                        writeOutput( "out1_Sub", out1_Sub );
                        writeOutput( "out0_Rel", out0_Rel );
                }else
                {
                        out1_Sub=0;
                        writeOutput( "out1_Sub", out1_Sub );
                }


                if(var2f<-1)
                {
                        out2_Baj=1;   ;
                        out0_Rel=1;
                        writeOutput( "out2_Baj", out2_Baj );
                        writeOutput( "out0_Rel", out0_Rel );
                }else
                {
                        out2_Baj=0;
                        writeOutput( "out2_Baj", out2_Baj );
                }

/*Condición de Activación (Apertura y Cierre del Gripper)*/

                if(var2f<1 && var2f>-1)
                {
                        out0_Rel = 0;
                        writeOutput( "out0_Rel", out0_Rel );
                        out0_Rel = in0_Act;
                        out3_Gri = in2_Gri;
                        writeOutput( "out0_Rel", out0_Rel );
                        writeOutput( "out3_Gri", out3_Gri );
                }

                writeOutput( "out4_Pos", out4_Pos );
        };

        writeOutput( "out0_Rel", out0_Rel );
        writeOutput( "out3_Gri", out3_Gri );
}
```

Fuente: Microsoft Visual Studio 2010.
Elaborador por: Cristian Cuji.

Generando la solución (F6) en Visual Studio 2010, se traduce el código fuente al nuevo bloque de función. El resultado final es obtener el control del Módulo Manipulador de objetos desde RobotinoView, programando en lenguaje gráfico y grafset.

3.4. Resumen de capítulo

El capítulo de Diseño, Desarrollo e Implementación explica y detalla la elaboración del proyecto en sus tres áreas: La construcción del sistema Mecánico, La construcción del sistema de control electrónico, La construcción del sistema driver de control y la

finalmente la implementación de los tres sistemas conformando un único sistema llamado Manipulador de Objetos.

El primer subcapítulo presenta datos técnicos, medidas y características del sistema mecánico. Pero al no ser el tema principal de estudio no se presenta mayores detalles de la elaboración del módulo. Lo más importante es obtener un sistema mecánico que pueda ser controlado con un sistema electrónico.

El segundo subcapítulo presenta con mayor profundidad información de características, funcionamientos, planos esquemáticos, algoritmos de programación y todo lo correspondiente al sistema electrónico de control. Se dedico mayor énfasis en explicar el funcionamiento y los diferentes elementos electrónicos presentes en la tarjeta de control.

El tercer subcapítulo es la implementación de los dos sistemas anteriores; Mecánico y Electrónico. En un dispositivo driver de control desarrollado con la herramienta RobotinoView_API_v2.8.4 que es casi la última versión en ser publicada por FESTO. Este subcapítulo se presenta a mi forma personal el mayor aporte de mi trabajo de investigación, ya que las herramientas API son poco conocidas pero poseen un amplio campo para aplicaciones y desarrollo de nuevas soluciones.

En este punto el módulo manipulador de objetos cumple con las características de cualquier módulo didáctico:

- Totalmente modular al Robotino.
- Adaptable según nuevas aplicaciones.
- Programable por medio de un bloque de funciones.
- Fácil de incorporar al chasis del Robotino.

CAPÍTULO 4
ANÁLISIS DE RESULTADOS

A continuación se presentan los resultados y descripción de pruebas de funcionamiento realizadas al módulo manipulador de objetos, principalmente la exactitud de las posiciones en las que se puede ubicar el gripper. Con el objetivo de demostrar que el módulo posee un funcionamiento óptimo. Para esto se ha realizado una serie de ejercicios de repetitividad con el objetivo de medir los rangos de error que se pueden producir en el sistema.

4.1. Pruebas de Repetitividad

4.1.1. Condiciones Ideales

Tabla- 16. Ubicación y Posición referenciales

Posición	Ubicación	Descripción
-1	0,1	Punto final Inferior
0	0,5	Capturar del piso 1
1	1	Capturar del piso 2
2	3,5	Posición elevada 1
3	5	Posición elevada 2
4	6,5	Posición elevada 3
5	8	Posición elevada 4
6	9,5	Posición elevada 5
7	11	Posición elevada 6
8	12,5	Posición elevada 7
9	14	Posición elevada 8
10	14,5	Punto final Superior
Distancia Total Recorrida	14,5	cm

Elaborado por: Cristian Cuji.

La Tabla- 16. Ubicación y Posición referenciales, expresa un gráfico de Posición versus Ubicación, se ha considerado 12 posiciones diferentes, programadas a distintas alturas. Punto final Inferior y Superior son puntos exactos ubicados a 1mm y 14,5 cm del piso respectivamente.

Las posiciones 0 y 1 se encuentran muy pegadas al piso a 0,5cm y a 1 cm del piso, ambas posiciones están consideradas para capturar objetos del piso. Las posiciones 2 al

9 son posiciones de elevación que permiten alcanzar diferentes alturas a las que se pueden depositar o capturar objetos.

4.1.2. Prueba general de posiciones

La prueba general consiste en recorrer 10 veces al gripper por toda la longitud del tornillo sin fin, ubicando al gripper en cada posición establecida en la tabla de condiciones iníciales dividiendo la prueba en dos grupos de información:

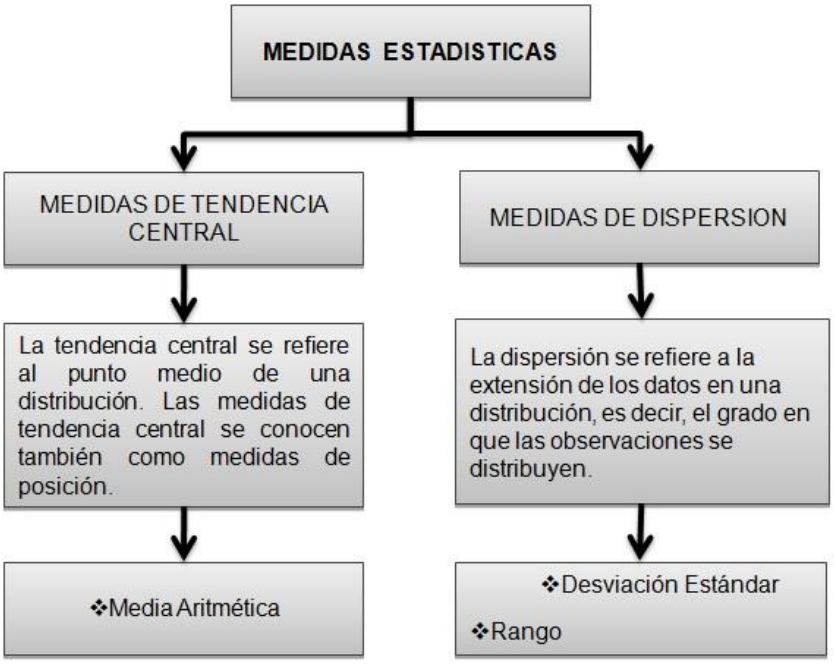
- Error producido en Subida.
- Error producido en Bajada.

En cada posición se registra el valor o rango de error, tanto en subida como en bajada. Una vez recogido los datos se procedió a realizar cálculos de Error Típico, Desviación Estándar, Media y Rango.

4.1.3. Análisis estadístico

Para el análisis se ha considerado aplicar dos tipos de datos o medidas, estas son las medidas de tendencia central y de dispersión. (GUTIERREZ PULIDO & DE LA VARA SALAZAR, 2008)

Fig. 66. Medidas Estadísticas



Elaborado por: Cristian Cuji.

4.1.3.1. Media aritmética

Para obtener la Media Aritmética se procedió a sumar todas las pruebas de posicionamiento del gripper y dividir la suma total con el número de pruebas realizadas:

$$X = \frac{\sum X}{N}$$

4.1.3.2. Desviación estándar

Con la desviación estándar se puedo conocer con exactitud la dispersión producida en cada posición con respecto al promedio.

$$\sigma = \sqrt{\frac{(X - \bar{X})^2}{N}}$$

4.1.3.3. Error estándar

Con ayuda de la desviación estándar y la media aritmética se pudo localizar en que punto de las pruebas de repetición empieza a existir una dispersión de las posiciones desde la media aritmética.

$$\sigma\bar{x} = \frac{\sigma}{\sqrt{n}}$$

4.1.3.4. Rango

Es la diferencia entre el dato mayor y el dato menor, encontrado en una misma posición, pero que se ve afectada por la lectura del sensor en subida o en bajada.

$$R = Xmax - Xmin$$

Las medidas mencionadas son principalmente las que se presentan en los siguientes cuadros, Tabla- 17. Cuadro de Error en Bajada Y Tabla- 18. Cuadro de Error en Subida. Los datos más relevantes para el funcionamiento del sistema son el Error Estándar y el Rango. Ambas medidas señalan los puntos de posición donde podrían existir errores o fallas en el funcionamiento.

4.1.4. Error producido en bajada

Tabla- 17. Cuadro de Error en Bajada

EN BAJADA															
Posición	**Ubicación**	A	B	C	D	E	F	G	H	I	J	MEDIA	DESVIACIÓN ESTANDAR	ERROR TÍPICO	RANGO
-1	**0,1**	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,000%	0,000%	0
0	**0,5**	0,5	0,5	0,5	0,5	0,6	0,5	0,5	0,5	0,5	0,6	0,51818	4,045%	1,220%	0,1
1	**1**	1,2	1,2	1,2	1,2	1,2	1,1	1,2	1,2	1,1	1,1	1,15455	6,876%	2,073%	0,2
2	**3,5**	3,6	3,7	3,6	3,6	3,6	3,6	3,7	3,6	3,6	3,7	3,61818	6,030%	1,818%	0,2
3	**5**	5,1	5,1	5,2	5,2	5,2	5,2	5,1	5,1	5,2	5,1	5,13636	6,742%	2,033%	0,2
4	**6,5**	6,5	6,4	6,5	6,4	6,4	6,4	6,4	6,4	6,4	6,4	6,42727	4,671%	1,408%	0,1
5	**8**	8,1	8,1	8	8	8	8,1	8,1	8,1	8	8	8,04545	5,222%	1,575%	0,1
6	**9,5**	9,5	9,5	9,5	9,5	9,5	9,6	9,5	9,5	9,6	9,5	9,51818	4,045%	1,220%	0,1
7	**11**	11	11	11	11,1	11	11,1	11	11	11	11	11,0182	4,045%	1,220%	0,1
8	**12,5**	12,6	12,6	12,5	12,5	12,5	12,5	12,5	12,5	12,5	12,5	12,5182	4,045%	1,220%	0,1
9	**14**	14	14	14	14	14,1	14,1	14	14	14	14	14,0182	4,045%	1,220%	0,1
10	**14,5**	14,5	14,5	14,5	14,5	14,5	14,5	14,5	14,5	14,5	14,5	14,5	0,000%	0,000%	0

Elaborado por: Cristian Cuji.

4.1.5. Error producido en subida

Tabla- 18. Cuadro de Error en Subida

EN SUBIDA															
Posición	**Ubicación**	A	B	C	D	E	F	G	H	I	J	MEDIA	DESVIACIÓN ESTANDAR	ERROR TÍPICO	RANGO
-1	**0,1**	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,000%	0,000%	0
0	**0,5**	0,5	0,6	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,50909	3,015%	0,909%	0,1
1	**1**	0,8	0,8	0,8	0,7	0,9	0,8	0,8	0,8	0,7	0,7	0,8	8,944%	2,697%	0,3
2	**3,5**	3,1	3,2	3,2	3,2	3,1	3,2	3,2	3,2	3,2	3,2	3,20909	10,445%	3,149%	0,4
3	**5**	4,3	4,2	4,3	4,3	4,3	4,3	4,3	4,2	4,3	4,2	4,33636	22,482%	6,779%	0,8
4	**6,5**	6,2	6,3	6,2	6,3	6,3	6,3	6,3	6,3	6,2	6,3	6,29091	8,312%	2,506%	0,3
5	**8**	7,7	7,6	7,7	7,7	7,5	7,5	7,7	7,6	7,6	7,6	7,65455	13,685%	4,126%	0,5
6	**9,5**	9,5	9,4	9,5	9,5	9,5	9,5	9,5	9,5	9,5	9,4	9,48182	4,045%	1,220%	0,1
7	**11**	10,9	11	11	11	10,9	11	11	11	11	11	10,9818	4,045%	1,220%	0,1
8	**12,5**	12,5	12,5	12,5	12	12,5	12,5	12,5	12,5	12,5	12,5	12,4818	6,030%	1,818%	0,2
9	**14**	14	14	13,9	14	13,9	14	14	14	14	14	13,9818	4,045%	1,220%	0,1
10	**14,5**	14,5	14,5	14,5	15	14,5	14,5	14,5	14,5	14,5	14,5	14,5	0,000%	0,000%	0

Elaborado por: Cristian Cuji.

4.1.6. Análisis de repetibilidad

"Según (CREUS, 7ma edición), La repetibilidad es sinónimo de precisión". Es decir si la repetibilidad es frecuente la dispersión de valores es menor por lo tanto existe mayor precisión.

CREUS, también plantea la siguiente fórmula para interpretar la repetibilidad:

$$\sqrt{\frac{\sum(xi - x)^2}{N}}$$

Donde:

x_i.- es el valor ingresado (Set point).

x.- es el valor real resultado del control.

N.- número de muestras.

Tomando los valores obtenidos durante el ejercicio de repetibilidad de ascenso y descenso se obtuvieron los datos presentados en la Tabla- 18. Cuadro de Error en Subida Y Tabla- 17. Cuadro de Error en Bajada Los datos obtenidos en todas las ubicaciones podrían ser analizados hasta 10 veces que es el número de ejercicios realizados en las tablas antes mencionadas.

En resumen a continuación, se realiza el análisis de repetibilidad para dos puntos diferentes en cada ejercicio, tomando los ejercicios con las ubicaciones más representativas.

Tabla- 19.- Datos de Repetibilidad en Subida

Subida			
Valor Xi	Valor X	Xi - X	(Xi - X)2
0,1	0,1	0	0,000
0,5	0,5	0	0,000
0,8	0,7	0,1	0,010
3,2	3,2	0	0,000

4,3	4,3	0	0,000
6,2	6,3	-0,1	0,010
7,7	7,7	0	0,000
9,5	9,5	0	0,000
11	11	0	0,000
12,5	12,3	0,2	0,040
13,9	14	-0,1	0,010
14,5	14,5	0	0,000
Suma de diferencias de cuadrados			0,070
Suma de diferencias de cuadrados/N			0,006
Raíz cuadrada (Suma total de cuadrados de las diferencias/N)			0,076376262

Subida			
Valor Xi	Valor X	Xi - X	(Xi - X)2
0,1	0,1	0	0,000
0,5	0,5	0	0,000
1	0,7	0,3	0,090
3,5	3,2	0,3	0,090
5	4,3	0,7	0,490
6,5	6,3	0,2	0,040
8	7,7	0,3	0,090
9,5	9,5	0	0,000
11	11	0	0,000
12,5	12,3	0,2	0,040
14	14	0	0,000
14,5	14,5	0	0,000
Suma de diferencias de cuadrados			0,840
Suma de diferencias de cuadrados/N			0,070
Raíz cuadrada (Suma total de cuadrados de las diferencias/N)			0,26457513

Elaborado por: Cristian Cuji.

Del primer análisis de repetibilidad se realiza con los ejercicios en subida o ascenso, donde se obtiene un valor de 0,076 y un valor de 0,264, que son valores alejado del 1 lo cual nos indica una precisión buena y aceptable para el sistema.

Tabla- 20.- Datos de Repetibilidad en Bajada

Bajada			
Valor Xi	Valor X	Xi - X	(Xi - X)2
0,1	0,1	0	0,000
0,5	0,5	0	0,000
1,2	1,2	0	0,000
3,7	3,6	0,1	0,010
5,1	5,2	-0,1	0,010
6,4	6,5	-0,1	0,010
8,1	8	0,1	0,010
9,5	9,5	0	0,000
11	11	0	0,000
12,6	12,5	0,1	0,010
14	14	0	0,000
14,5	14,5	0	0,000
Suma de diferencias de cuadrados			0,050
Suma de diferencias de cuadrados/N			0,004
Raíz cuadrada (Suma total de cuadrados de las diferencias/N)			0,06454972

Bajada			
Valor Xi	Valor X	Xi - X	(Xi - X)2
0,1	0,1	0	0,000
0,5	0,5	0	0,000
1	1,2	-0,2	0,040
3,5	3,6	-0,1	0,010
5	5,2	-0,2	0,040
6,5	6,5	0	0,000
8	8	0	0,000
9,5	9,5	0	0,000
11	11	0	0,000
12,5	12,5	0	0,000
14	14	0	0,000
14,5	14,5	0	0,000
Suma de diferencias de cuadrados			0,090
Suma de diferencias de cuadrados/N			0,008
Raíz cuadrada (Suma total de cuadrados de las diferencias/N)			0,08660254

Elaborado por: Cristian Cuji.

El segundo análisis se realiza en los ejercicios de bajada o descenso donde se obtiene valores de 0,064 y 0,086 valores bastante alejados del 1, lo cual nos indica una gran precisión, incluso mayor al ejercicio de subida.

Entonces aplicando la formula de repetibilidad en los siguientes puntos se obtuvo los siguientes resultados.

4.1.7. Interpretación de datos

Los resultados de las pruebas de funcionamiento aplicando análisis estadístico se dividen en dos partes: Análisis de resultados en subida y en bajada.

La razón de la división en el análisis estadístico es porque el sensor encargado de controlar la posición tiene un comportamiento irregular en algunas posiciones produciendo errores en la mayoría de casos en subida. Durante el descenso del gripper el error no es muy notorio porque se utilizo las posiciones de descenso para calibrar el sistema.

La conclusión que se deber tener en cuenta es que el sistema reacciona de mejor manera durante del ejercicio de bajada o descenso, es decir posee una mayor precisión. Mientras que en el ejercicio de subida posee una menor precisión.

4.1.7.1. Interpretación de datos en subida

Los datos en subida son los que poseen mayor medida en el rango de variación entre el punto máximo y el mínimo, esto es evidente en las posiciones 2, 3, 4 y 5. Donde el rango es superior a 0,3cm. Inclusive existe un error crítico en la posición 3 de 8mm de distancia entre el valor deseado y el valor donde se fija el gripper. Las demás posiciones no presentan mayores rangos de error. Y son generalmente posiciones muy exactas.

4.1.7.2. Interpretación de datos en bajada

En bajada no existen rangos mayores a 2mm, entre la posición deseada y la posición alcanzada por el gripper. Esto es ocurre porque el gripper fue calibrado durante la secuencia de descenso o bajada.

Se debe recalcar que los valores de punto final superior e inferior son valores exactos porque además del sensor de posición, en estas posiciones se hace un control con los sensores final de carrera. Por este motivo casi no presentan error o rango.

Las posiciones más cercanas al sensor de distancia, entre ellas la posición 10, 9, 8, 7, 6 y 5 son posiciones que poseen el menos grado de error o rango. Esto ocurre ya que el funcionamiento del sensor es casi lineal mientras más cercano se encuentre el gripper.

4.2. Costos de la investigación

Los costos de la investigación se presentan a continuación en la Tabla- 21. Costos de la Investigación

Tabla- 21. Costos de la Investigación

Item	Fecha	Nº de Factura	Descripción	Empresa	Cantidad	Valor Unitario($ Dolares)	Valor ($ Dolares)
1	10/02/2012	0	Tornillo sin Fin	Mecánica Industrial	1	25,00	25,00
2	10/02/2012	0	Elementos Mecanicos	Mecánica Industrial	1	100,00	100,00
3	21/05/2012	20493	Motor de Pasos	Selectronic	1	7,14	7,14
4	24/05/2012	4	Microcontrolador PIC 18F2550	Megatrónica	1	9,50	9,50
5	24/05/2012	4	Pulsador	Megatrónica	1	0,17	0,17
6	24/05/2012	4	Optoacoplador PC817	Megatrónica	8	0,55	4,40
7	24/05/2012	4	Cristal 12MHz	Megatrónica	1	0,59	0,59
8	24/05/2012	4	Tip 122	Megatrónica	4	0,60	2,40
9	24/05/2012	4	Resistencias	Megatrónica	40	0,02	0,80
10	24/05/2012	4	Regulador 7805	Megatrónica	1	0,49	0,49
11	26/07/2012	1107	Baquelita 10 x 20	Megatrónica	1	1,45	1,45
12	26/07/2012	1107	Acido	Megatrónica	2	1,25	2,50
13	26/07/2012	1107	Borneras	Megatrónica	30	0,30	9,00
14	13/08/2012	9965	Elementos Electrónicos	Omega	1	7,40	7,40
15	10/10/2012	61	Elementos Electrónicos	Megatrónica	1	7,24	7,24
16	24/10/2012	80824	Tornillos de Maquina	MegaKywi	1	3,73	3,73
17	22/12/2012	0	Entorno de madera	Carpinteria	1	100,00	100,00
18	15/01/2012	94874	Spray Aluminio	MegaKywi	1	11,32	11,32
19	03/01/2013	11256	Bateria Seca 12V-2Amp	Electronica del Norte	2	11,90	23,80
						IVA 12%	38,03
						SubTotal	278,90
						TOTAL	316,93

Elaborado por: Cristian Cuji.

CONCLUCIONES Y RECOMENDACIONES

CONCLUSIONES

- Robotino View v2.8.4 proporciona las herramientas necesarias para desarrollar iconos personalizados, generando nuevas aplicaciones para el Robot Móvil Robotino. Se debe recalcar que el instalador de las herramientas debe ser de la misma versión de Robotino View es decir: RobotinoView_API_2.8.4.
- RobotinoView_API_2.8.4 facilita las herramientas: Function Block Manager y CMaker-GUI. Que son las herramientas necesarias para desarrollar el nuevo bloque de funciones generando un proyecto en un software compilador por ejemplo Visual Studio 2010.
- Function Block Manager permite construir físicamente el nuevo icono o bloque de funciones, asignando el número de entrada, número de salidas, tipo de dato, variables, gráfico del icono y dirección. Además genera un código fuente que puede ser traducido a un software compilador donde el nuevo bloque de funciones puede ser editado.
- CMaker-Gui traduce el código fuente desarrollado por Function Block Manager a un lenguaje de compilador seleccionado, generando un código fuente que puede ser editado y nuevamente cargado al nuevo bloque de funciones. El compilador se utiliza exclusivamente para genera una solución al código (F6), nunca se debe compilar el código (F9) ya que la compilación genera errores propios del código generado por Function Block Manager.
- La arquitectura mixta de SISC y RISC establece una división de tareas, para el proyecto SISC (Computador del Robotino) y RISC (Tarjeta de control electrónico). Claramente se encuentran divididas las tareas. Robotino se encarga de procesar grandes cantidades de información por ejemplo: Wirelless, Sensores, Camara Web, Motores o ruedas omnidireccionales, entradas y salidas, Etc. La tarjeta de control diseñada para el proyecto procesa un conjunto reducido de información y prácticamente solo responde a las instrucciones del sistema SISC.
- Un microcontrolador Microchip nunca deja de ser vulnerable al ruido electromagnético, ya que el ruido siempre está presente gracias a diferentes

fuentes como motores, señales de teléfonos celulares, estática producida por ropa de poliéster, etc. Lo recomendable es proteger al microcontrolador del ruido por medio de: Malla de tierra, capacitores, filtros y diferentes tipos de técnicas para proteger al dispositivo.

- La fuente de alimentación es un detalle muy importante en todos los sistemas electrónicos, obtener una fuente constante de corriente y voltaje sin grandes variaciones es importante para evitar que el Microcontrolador caiga en un reinicio indeseado. Aunque se utilice diodos zenner, diodos rectificadores, reguladores de voltaje o convertidores, si la fuente principal "Batería de $12V_{DC}$" no abástese un voltaje superior al nominal de $5V_{DC}$, necesarios para alimentar el circuito de control ninguna técnica conocida funcionara. Por esta razón el proyecto cuenta con una batería y un cargador con especificaciones precisas para el proyecto desarrollado.
- Y finalmente la programación del Microcontrolador también colabora con el sistema electrónico para evitar problemas de funcionamiento, por ejemplo evitar puntos de reinicio indeseados. Cada vez que ocurra un reinicio indeseado "Reset" el microcontrolador entra a funcionar no desde el punto inicial, por el contrario ingresa directamente a leer los datos y a ejecutar acciones directamente, para el proyecto este algoritmo disminuyo considerablemente el rango de errores.
- La mecánica fue una debilidad para el desarrollo de proyecto, porque los conocimientos y la técnica necesaria para desarrollar un sistema de cuerpo rígido, dependían de terceras personas, específicamente un mecánico industrial. El diseño en AutoCad facilito considerablemente un entendimiento entre el mecánico y el objetivo del tema del proyecto.
- La mayor investigación se enfoco en la búsqueda de un método de transmisión ideal de movimiento mecánico, sin ejercer mayor carga o esfuerzo al sistema y que sea técnicamente posible desarrollarlo por un mecánico, la solución fue un sistema de tornillos sin fin y un sistema de engranajes de relación 1 a 1. Además se conto con un mecanismo de pinza o gripper diseñado en plástico que fue fácilmente adaptado.

RECOMENDACIONES

- Para utilizar las herramientas proporcionadas por el API (Application Programming Interface), se recomienda descargase los instaladores de la misma versión, caso contrario el sistema no va a funcionar correctamente.
- Los manuales proporcionados por Festo Didactic, son claros en todas las especificaciones de configuración del API_Robotino. Sin embargo en mi desarrollo de investigación presento un manual con un detalle mayor de información para la configuración inicial del API.
- El sistema funciona de forma correcta con un voltaje de batería de entre $13V_{DC}$ a $8V_{DC}$, por debajo del voltaje recomendado el módulo empezara a tener atrancamientos producidos por la falta de torque en el motor de pasos.
- Se recomienda no desmontar el motor y el sensor de posicionamiento porque son dispositivos que fueron calibrados y al ser desmontados pueden generar errores.
- La tarjeta de control electrónica está diseñada para funcionar con cualquier microcontrolador de 24 pines de gama media y alta. Por ejemplo Pic18F2550 o Pic16F877.
- El lenguaje de programación C++ debería ser desarrollado con mayor énfasis en la formación académica porque la mayoría de API (Application Programming Interface), se encuentran disponibles en leguaje: C, C++ o C#. No se debe olvidar que los API son proporcionados por diferentes marcas de dispositivos electrónicos para hacer exclusivamente desarrollo e investigación.

BIBLIOGRAFÍA

ALCIATORE, D. G., HISTAND, M. B., & CASTAÑEDA Cedeño, S. (2008). *INTRODUCCIÓN A LA MECATRÓNICA Y LOS SISTEMAS DE MEDICIÓN.* Colorado - U.S.A.: Mc Graw Hill.

ANGULO USATEGUI, J. M., & ROMERO. (2da edición 2006). *MICROCONTROLADORES PIC "DISEÑO PRACTICO DE APLICACIONES".* Madrid: McGrawHill.

BARRIENTOS, PEÑIN, & BALAGUER. (2007.). *"Fundamentos de Robótica".* Madrid: McGraw-Hill.

BATES, M. (2006). *INTERFACING PIC MICROCONTROLLERS - EMBEDDED DESIGN BY INTERACTIVE SIMULATION.* Amsterdam: ELSEVIER - NEWNES.

BEER P, F., JOHNSTON E, R., & EISENBERG R, E. (2005). *MECÁNICA VECTORIAL PARA INGENIEROS - ESTÁTICA.* Madrid: Mc Graw Hill.

BISHOP, O. (2007). *THE ROBOT BUILDER´S COOKBOOK.* ANSTERDAN: ELSEVIER NEWNES.

CREUS, A. (7ma edición). *Instrumentación Industrial*. Madrid.

DICCIONARIO DE LA LENGUA ESPAÑOLA 10maE. (2012). Real Academia de Lengua Española. *DICCIONARIO DE LA LENGUA ESPAÑOLA - Vigésima segunda edición* .

FESTO Didactic GmbH & Co. KG. (2008). *Manual Robotino Nº 564176.* Alemania: Festo ©.

FESTO, A. (2012). *COLE_DIDACTICUM_CIA_LTDA*. Recuperado el 20 de FEBRERO de 2012, de http://www.festo.com/net/startpage/

FESTO, B. W. (2007). *FESTO Robotino - Manual del Instructor.* Denkendorf: FESTO Didactic GmbH & Co. KG.

FITZGERALD, A. E., KINGSLEY, C. J., & UMANS, S. D. (2004). *MÁQUINAS ELÉCTRICAS.* Massachusetts - U.S.A: Mc Graw Hill - Interamericana.

GUTIERREZ PULIDO, H., & DE LA VARA SALAZAR, R. (2008). *ANALISIS Y DISEÑO DE EXPERIMENTOS.* Mexico D.F.: McGraw Hill.

GUTIÉRREZ SALAZAR, F. E., & CRUZ LAVERDE, J. L. (2008). *AutoCAD 2008 - 3D Y ESPACIO PAPEL.* Bogotá, D.C.: Alfaomega Colombiana S.A.

IBRAHIM, D. (September, 2002). *MICROCONTROLLES BASED TEMPERATURE MONITORING AND CONTROL.* U.S.A.: Newnes.

ISO_8373. (1998). Robot Industrial Manipulador. *Norma UNE EN ISO 8373* , 1.

JENSEN, C., SHORT, D. R., & HELSEL, J. D. (2002). *DIBUJO Y DISEÑO EN INGENIERÍA.* México, D.F.: McGraw-Hill Interamericana.

JOYANES AGUILAR, L., & ZAHONERO MARTÍNEZ, I. (2010). PROGRAMACIÓN en C, C++, JAVA y UML. Madrid: Mc Graw Hill.

LINDSAY, A. (2005). *WHAT´S A MICROCONTROLLER?* U.S.A.: PARALLAX INC.

MATIC, N. (January 2003). *BASIC FOR PIC MICROCONTROLLERS.* U.S.A.: Mikroeletronica C.O.

MATIC, N. (May 15, 2000). *PIC MICROCONTROLLERS BOOK1.* U.S.A.: Mikroelektronika C.O.

ORÚS ASSO, F. (1977). En F. ORÚS ASSO, *MATERIALES DE CONSTRUCCIÓN* (pág. 365 y 366). Madrid: DOSSAT, S.A. 7ma Edición.

ORÚS ASSO, F. (1977). *MATERIALES DE CONSTRUCCIÓN.* Madrid: DOSSAT S.A., 7ma Edición.

PREDKO, M. (2002). *PROGRAMMING AND CUSTOMIZING THE PIC MICROCONTROLLER.* NEW YORK: TAB ELECTRONICS.

PREDKO, M. (2009). *PROGRAMMING AND CUSTOMIZING THE PIC MICROCONTROLLER.* New York - U.S.A.: Mc Graw Hill.

Real Academia de la Lengua Española. (2001). *Diccionario.* Madrid: 22º Edicion.

RIA. (2005). Definición de Robot. *RIA (Robot Institute of America, actualmente Robotic Industries Association)* , 1.

ROLDÁN VILORIA, J. (2002). *PRONTUARIO DE MECÁNICA INDUSTRIAL APLICADA.* Madrid: Thomson - Paraninfo.

SEFARAD, I. (2010). *Instituto de Educacion Superior SEFARAD.* Toledo: Departamento de Tecnologia.

SMITH, J. R. (2005). *PROGRAMMING THE MICROCONTROLLER WITH MBASIC.* Amsterdam: ELSEVIER - NEWNES.

TORRES POMARES, G., & PUENTE, A. (2005). *ROBOTS Y SISTEMAS SENSORIALES.* Madrid: Prentice Hall.

NETGRAFÍA

WEB_010. (12 de Marzo de 2012). *Enciclopedia Libre "Wikipedia"*. Recuperado el 15 de Marzo de 2012, de Arquitectura ARM: http://es.wikipedia.org/wiki/Arquitectura_ARM

WEB_010. (27 de Febrero de 2012). *La Enciclopedia Libre; Wikipedia*. Recuperado el 10 de Marzo de 2012, de RISC: http://es.wikipedia.org/wiki/RISC

WEB_011. (17 de Octubre de 2011). *La Enciclopedia Libre; Wikipedia*. Recuperado el 08 de Marzo de 2012, de CISC: http://es.wikipedia.org/wiki/CISC

WEB_013. (20 de Noviembre de 2011). *Robots móviles (y VII)*. Recuperado el 31 de marzo de 2012, de Robotica: http://www.xatakaciencia.com/robotica/robots-moviles-y-vii

WEB_016. (Julio de 2011). Recuperado el 23 de Abril de 2012, de DRAMS Tecnology S.A.: http://www.dramstechnology.com.ar/productos.php?prodid=99

www.aprendemostecnologia.org. (2008). *Aprendamos Tecnología*. Recuperado el 2012 de Abril de 28, de http://aprendemostecnologia.org/maquinas-y-mecanismos/mecanismos-de-transmision-del-movimiento/

www.assig.fib.upc.es. (2005). *Grafico Teatro de Herón de Alejandría*. Recuperado el 17 de enero de 2012, de Robots de Inspiración Biológica: http://www-assig.fib.upc.es/~rob/protegit/treballs/Q2_03-04/insp_biolog/Robots%20de%20inspiraci%F3n%20biologica.htm

www.automata.cps.unizar.es. (2004). *Grafico de el Gallo de Estrasburgo*. Recuperado el 17 de enero de 2012, de Autómatas en la Historia: http://automata.cps.unizar.es/Historia/Webs/automatas_en_la_historia.htm

www.blogspot.com/crfzoraida. (21 de junio de 2010). *ZAMBRANO Morales, Yenny Zoraida*. Recuperado el 22 de abril de 2012, de Blogspot, Metales pesados: http://crfzoraida.blogspot.com/2010/06/los-metales-metales-pesados-ambiente-y.html

www.blogspot.com/ignaciarcz. (10 de Junio de 2010). *Zenteno, Ignacia Carvallo*. Recuperado el 22 de Abril de 2012, de Blogspot, Evolución de los Materiales: http://ignaciarcz.blogspot.com/

www.cnnexpansion.com. (17 de Octubre de 2011). *CNNexpansion*. Recuperado el 18 de Febrero de 2012, de Articulo CNN “Un Robot me ganó el trabajo”: http://www.cnnexpansion.com/manufactura/2011/10/17/un-robot-me-gano-el-trabajo

www.directindustry.es. (22 de febrero de 2012). *Direct Industry*. Recuperado el 23 de Abril de 2012, de El Salón Virtual de la Industria: http://www.directindustry.es/fabricante-industrial/perfil-aluminio-60754.html

www.enciclopedia.cat. (2010). Recuperado el 28 de Abril de 2012, de Grup Enciclopedia Catalana: http://www.enciclopedia.cat/fitxa_v2.jsp?NDCHEC=0183172

www.es.wikipedia.org. (2012). *Wikipedia: La Enciclopedia Libre*. Recuperado el 15 de Marzo de 2012, de Arquitectura ARM: http://es.wikipedia.org/wiki/Arquitectura_ARM

www.es.wikipedia.org. (2010). *Wikipedia; La Enciclopedia Libre*. Recuperado el 23 de enero de 2012, de Lenguaje ensamblador: http://es.wikipedia.org/wiki/Lenguaje_ensamblador

www.es.wikipedia.org. (2010). *Wikipedia; La Enciclopedia Libre*. Recuperado el 23 de Enero de 2012, de Lenguajes de Programación: http://es.wikipedia.org/wiki/Lenguaje_de_alto_nivel

www.etimologias.dechile.net. (Agosto de 2006). *Etimologías*. Recuperado el 17 de febrero de 2012, de Significados Etimológicos: http://etimologias.dechile.net/?robot

www.historyofinformation.com. (agosto de 2006). *From Cave Paintings to the Internet*. Recuperado el 17 de enero de 2012, de Chronological and Thematic Studies on the History of Information and Media: http://www.historyofinformation.com/index.php?id=2306

www.interempresas.net. (2002). *Present and prospects of Robotics*. Recuperado el 18 de Febrero de 2012, de IFR (International Federation Of Robotics): http://www.interempresas.net/MetalWorking/Articles/1464-Present-and-prospects-of-Robotics.html

www.latiendadelmaestro.es. (2012). *La Tienda del Maestro*. Recuperado el 2012 de 04 de 28, de http://www.latiendadelmaestro.es/dibujo-tecnico/escuadras

www.refaccionesparacompresores.com. (28 de abril de 2011). *Metal Mecanica*. Recuperado el 20 de abril de 2012, de http://www.refaccionesparacompresores.com/refacciones-de-hierro-colado-para-compresores/

www.sitenordeste.com. (13 de Abril de 2012). Recuperado el 23 de Abirl de 2012, de Tecnología Mecánica: http://www.sitenordeste.com/mecanica/comercio_de_aceros.htm

www.wikipedia.org. (2006). *Wikipedia, La Enciclopedia Libre*. Recuperado el 17 de Febrero de 2012, de ROBOT: https://es.wikipedia.org/wiki/Robot

ANEXOS

ANEXOS Nº 1.- “BLOQUE DE FUNCIONES” PARA ROBOTINO VIEW 2.8.4

"BLOQUE DE FUNCIONES" PARA ROBOTINO VIEW 2.8.4

Cristian Cuji C

cristiancuji10@hotmail.com

Quito, febrero 2013

OBJETIVOS:

- Conocer que es API y su aplicación en proyectos de programación con el robot móvil Robotino.
- Generar información de configuración para desarrollar un nuevo bloque de funciones en Robotino View v2.8.4
- Conocer la aplicación que tiene las diferentes herramientas de software API en el desarrollo de un nuevo bloque de funciones.
- Desarrollar un ejemplo sencillo de la configuración de esta herramienta.

INTRODUCCIÓN:

Robotino View es un software de programación intuitivo en lenguaje grafico exclusivo para la programación del Sistema Robot Móvil Robotino de Festo.

Este manual describe el uso de la herramienta "Mis Bloques de Funciones" en Robotino View Versión 2.8.4 y RobotinoView_API 2.8.4.

Gráfico. 1.- Robotino - FESTO

Fuente: FESTO Didactic GmbH & Co. KG

¿Qué es un Bloque de Funciones?

Un Bloque de Funciones es un icono personalizado que puede ser introducido en RobotinoView v2.8.4. La programación se realiza en lenguaje C++ y es traducido al lenguaje grafico en RobotinoView por medio de un API. El bloque de funciones personalizado puede ejecutar tareas o algoritmos de acuerdo a las habilidades del programador.

¿Qué es API?

API (Application Programming Interface), una traducción aproximada puede ser Aplicación de Interfaz de Programación. El objetivo es tener acceso a librerías propias del Sistema Robot Móvil Robotino, por medio de un lenguaje de programación por ejemplo: C, C++, C#, Java, .NET, LabView, MatLab, entre otros.

API es una herramienta que proporciona la traducción entre lenguajes.

REQUERIMIENTOS:

Se requiere computador funcionando en sistema operativo Windows XP, Windows Vista, Windows 7 Home o Professional. Una conexión de internet inalámbrica. Se recomienda que el procesador sea superior o igual Intel Corel Dúo.

Y previamente ya se posea instalado con un compilador de Lenguaje C++ como por ejemplo: Eclipse, Borland C++, Ninja, Visual Studio 2007 o superior.

1. **Descargar el Software**

FESTO proporciona de forma gratuitita los instaladores, se los puede descargar de la siguiente página web:

http://doc.openrobotino.org/download/RobotinoView/index.php.

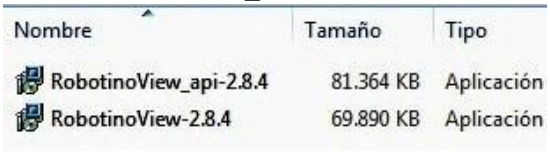
En la web mencionada se puede descargar Robotino View desde la versión 1.7.2., hasta la versión 2.9.4 y su respectivo RobotinoViewAPI para las mismas versiones. Además de encontrarse actualizaciones para la tarjeta de memoria del Robotino.

Para descargar el instalador del simulador de Robotino se recomienda visitar la siguiente web:

http://www.festo-didactic.com/int-es/learning-systems/robotino/?fbid=aW50LmVzLjU1Ny4xNC4yMC44NTg

Observación: La versión de RobotinoView y RobotinoViewAPI deben ser similares, caso contrario no va funcionar.

Gráfico. 2.- Instaladores RobotinoView y RobotinoView_API versión 2.8.4

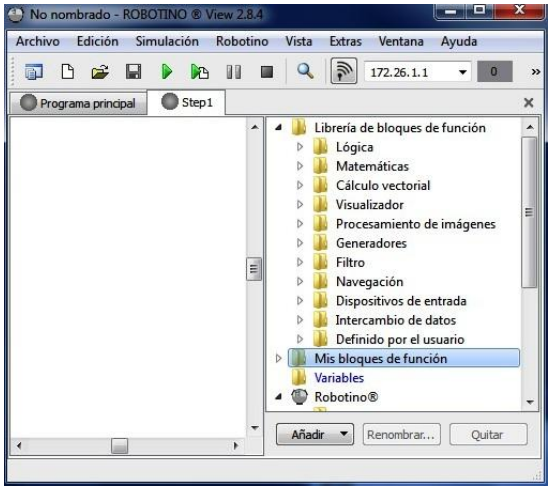


Elaborador por: Cristian Cuji.

2. Instalación

La instalación es sencilla solo damos doble clik, primero en RobotinoView-2.8.4 y finalmente en RobotinoView_api-2.8.4. Por medio de un asistente se instalará automáticamente ambos programas.

Gráfico. 3.- Entorno RobotinoView 2.8.4

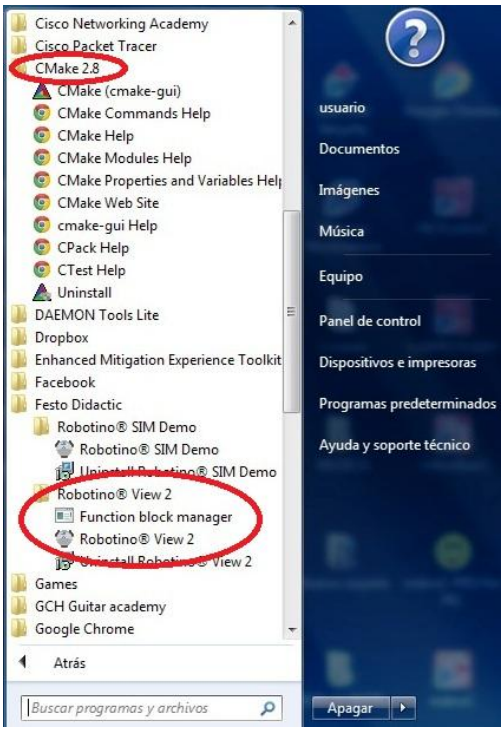


Elaborador por: Cristian Cuji.

Una vez que se termino el proceso de instalación, se puede verificar que los siguientes programas existan en la barra de inicio.

- Robotino View-2.8.4.
- Function Block Manager.
- CMake (cmake-gui)

Gráfico. 4.- Barra de Inicio, CMake, RobotinoView2 y Function Block Manager



Elaborador por: Cristian Cuji.

Adicionalmente puede instalarse el Simulador de Robotino.

3. Confirmar direcciones de carpetas de trabajo

Durante la instalación del software se crea automáticamente una carpeta por defecto de trabajo generalmente en una carpeta pública como por ejemplo: Usuario, Users, Documentos o alguna carpeta personalizada. Depende de la configuración de cada computador. En mi caso específico es la siguiente dirección:

C:\Users\usuario\Festo\RobotinoView2\units\MyFunctionBlocks

En esta dirección se encuentran los drives necesarios para el funcionamiento de los bloques de funciones.

Adicionalmente es necesario crear una nueva carpeta de trabajo, donde se van a editar los códigos de lenguaje C++, por ejemplo:

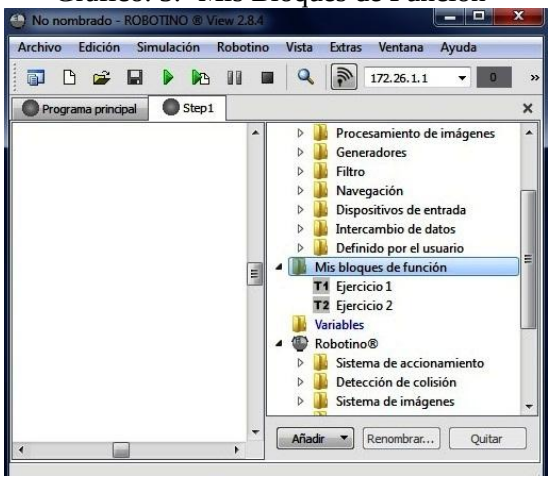
C:\ProyectoTesis\

En la nueva carpeta se van a editar los nuevos proyectos.

4. Vista Inicial

RobotinoView presenta la siguiente imagen cuando se inicia por primera vez, como se aprecia solo existen dos bloques de funciones: Ejercicio1 y Ejercicio2.

Gráfico. 5.- Mis Bloques de Función



Elaborador por: Cristian Cuji.

El objetivo es crear un nuevo icono personalizado, que es desarrollado en C++ y puede ser utilizado fácilmente en RobotinoView.

5. Function Block Manager

Esta herramienta se encuentra en la carpeta de Festo Didactic en la Barra de Inicio:

Gráfico. 6.- Barra de Inicio - Function Block Manager

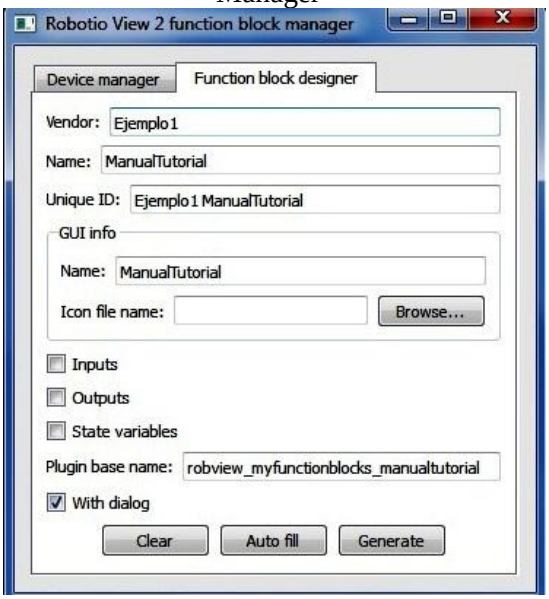
Elaborador por: Cristian Cuji.

Se instala al ejecutar el instalador del RobotinoView-API.

La función de esta herramienta es desarrollar o construir el nuevo bloque de funciones.

Primero ingresamos a la pestaña "Function Block Designer". En la ventana se llenan solo los datos de vendor y name, una vez llenados estos datos se da clik en "Auto Fill" y los demás datos se llenaran automáticamente.

Gráfico. 7.- Pantalla principal Function Block Manager

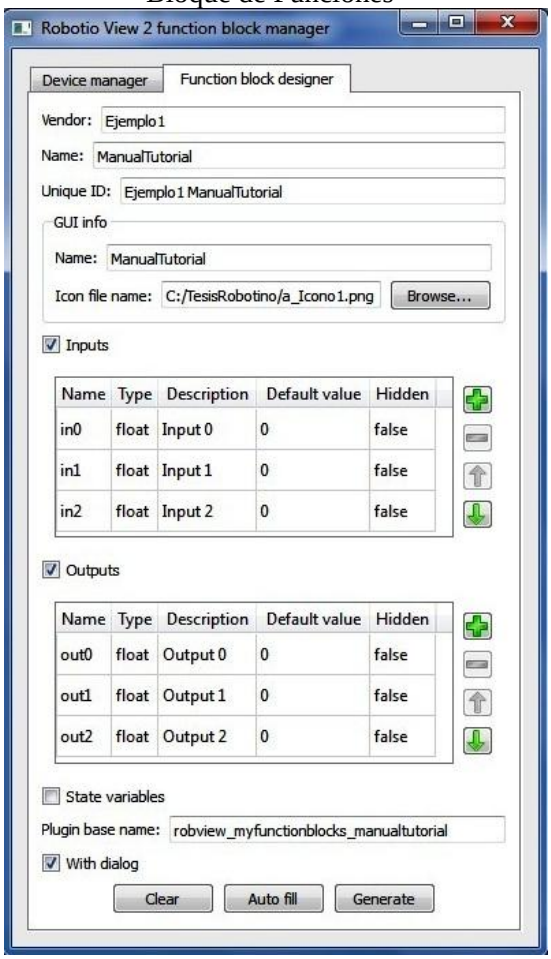


Elaborador por: Cristian Cuji.

En “Icon File Name” se despliega una venta de exploración Windows, aquí seleccionamos la dirección de un gráfico en extensión .PNG. El icono seleccionado será la imagen o gráfico del nuevo bloque de funciones.

Nota: Previamente el gráfico el debe ser modificado en Paint, Microsoft Office Picture Manager o algún editor de imágenes el tamaño ideal del grafico depende del proyecto y número de entradas, pero un tamaño ideal aproximado en pixeles es de 50 x 50 pixeles.

Gráfico. 8.- Configuración de un Nuevo Bloque de Funciones



Elaborador por: Cristian Cuji.

La misma herramienta permite establecer número de entradas, salidas y variables.

Al tratarse de un tutorial se va desarrollar un ejemplo de tres entradas y tres salidas. Activando “Outputs” o “Inputs” se puede agregar una salida, una entrada o una variable sencillamente dando clik en el símbolo +.

En la misma ventana se pueden modificar características como el tipo de variable, descripción, valor por defecto y hasta el propio nombre. Como un ejemplo a continuación modifico algunas de las características del nuevo bloque de funciones.

Se debe considerar muy bien qué tipo de variables se van a utilizar en el proyecto, una vez generado el proyecto las variables quedan definidas y el cambio de estas variables implica realizar nuevamente el bloque de funciones.

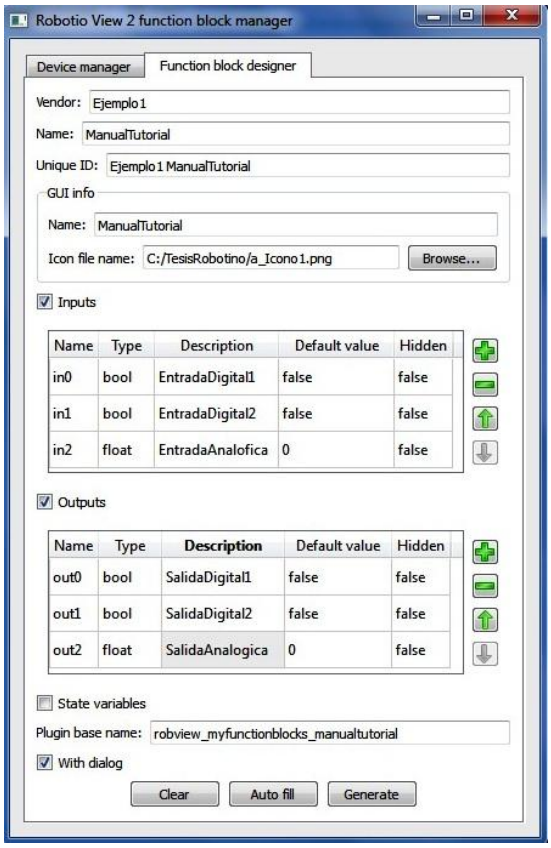
Las variables disponibles en esta versión son:

Gráfico. 9.- Tipos de Variables

TIPO	EJEMPLO
Float	0.00.... 100.55….
Int32	….- 100….0….100…
Unit32	0…..100…
Bool	true/false o 0/1

Elaborador por: Cristian Cuji.

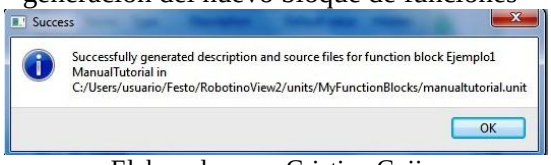
Gráfico. 10.- Edición del nuevo bloque de funciones



Elaborador por: Cristian Cuji.

Una vez establecidas las entradas y salidas del nuevo bloque de funciones se finaliza dando clik en "Generate" y el sistema nos devuelve un aviso de éxito o error.

Gráfico. 11.-Mensaje de éxito en la generación del nuevo bloque de funciones



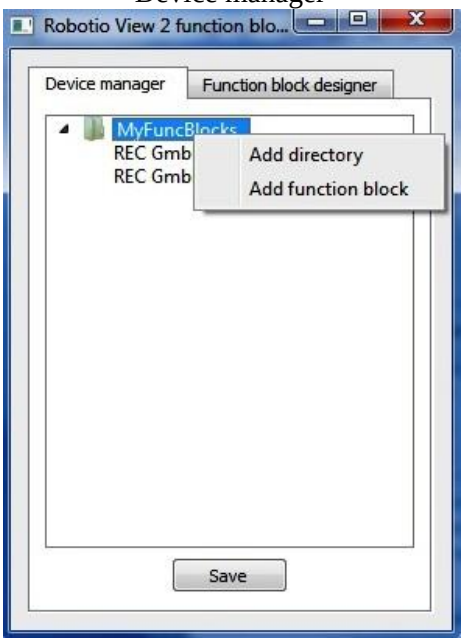
Elaborador por: Cristian Cuji.

Como se aprecia el nuevo proyecto se guardo en la antes mencionada carpeta de trabajo por defecto.

Para finalizar agregamos el nuevo bloque al RobotinoView. Esto se hace en la pestaña "Device Manager". Dando clik derecho sobre la carpeta "MyFuncBlock", se puede agregar un nuevo directorio o un nuevo bloque de funciones.

Por medio de este procedimiento el icono se vuelve transparente en RobotinoView2.8.4.

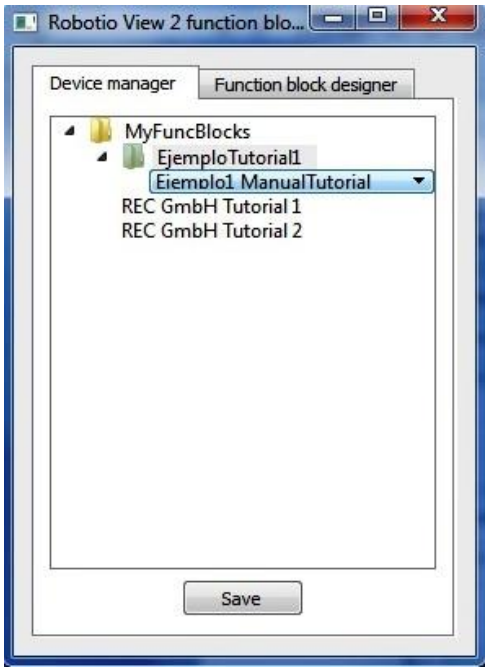
Gráfico. 12.- Function Block Manager - Device manager



Elaborador por: Cristian Cuji.

Agregado el nuevo proyecto se lo guarda dando clik en "Save" y se puede cerrar la herramienta "Function Block Manager"

Gráfico. 13.- Agregar nuevo bloque de funciones



Elaborador por: Cristian Cuji.

En este punto del tutorial se tiene un icono vacio, si el usuario intenta introducir el icono al entorno de programación gráfico de RobotinoView, se presentará un mensaje de error o advertencia.

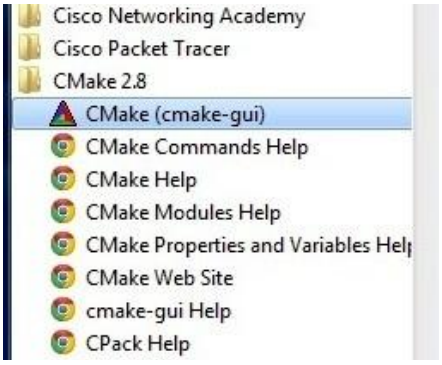
6. Cmake (cmake-gui)

CMake es otra herramienta proporcionada durante la instalación del API. Lo que hace CMake es traducir las variables designadas en la herramienta "Function Block Manager" al nuevo bloque de funciones en lenguaje C++, creando un proyecto traducido para ser modificado.

CMake interactúa con la carpeta por defecto y la carpeta de trabajo.

Se inicia dando clik en CMake (cmake-gui), presente en la barra de inicio.

Gráfico. 14.- Barra de Inicio - CMake



Elaborador por: Cristian Cuji.

En la venta principal se define dos direcciones "Browser Source" y "Browser Build". La primera es la carpeta por defecto, esta dirección varía de acuerdo a la configuración de cada computador pero generalmente son similares, en mi caso particular la dirección es:

C: /Users/usuario/Festo/RobotinoView2/units/MyFunctionBlocks/manualtutorial.unit/plugin

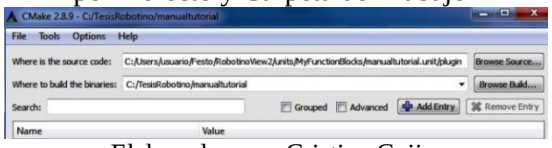
"PLUGIN" es una carpeta donde se encuentran el icono generado desde el "Function Block Manager", es decir contiene la información del nuevo bloque de funciones y conecta la programación de lenguaje C++ a lenguaje grafico.

La segunda dirección es una carpeta de trabajo, en esta carpeta es donde CMake genera el proyecto que puede ser editado.

C: /TesisRobotino/manualtutorial

Nota: El nombre del proyecto y de la nueva carpeta debe llevar el mismo nombre. En este caso es "manualtutorial"

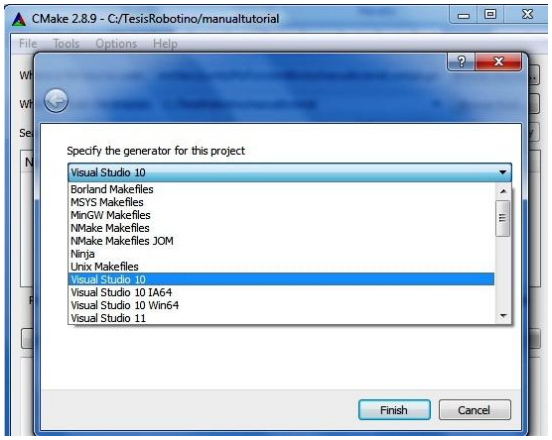
Gráfico. 15.- Cmake - Dirección de Carpeta por Defecto y Carpeta de Trabajo



Elaborador por: Cristian Cuji.

Con ambas direcciones especificadas, se procede a dar clik en configurar "Configure". El siguiente paso es seleccionar el compilador instalado previamente. Por ejemplo Visual Studio 2010.

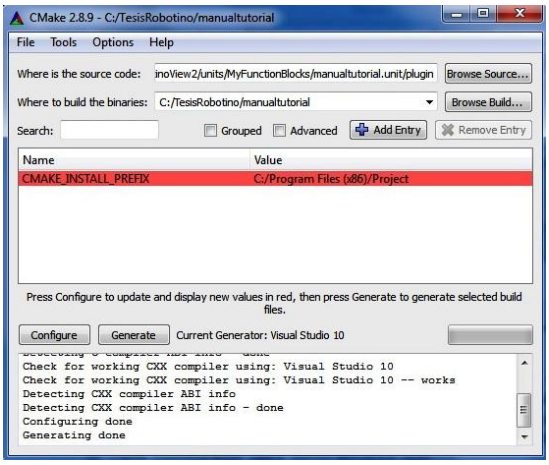
Gráfico. 16.- Selección del compilador de C++



Elaborador por: Cristian Cuji.

Clik en "Finish" para terminar la configuración, se espera hasta recibir el mensaje de "Done configure" y luego se procede da clik en generar "Generate".
La pantalla final en este paso es la siguiente:

Gráfico. 17.- Ventana de configuración CMake 2.8.4

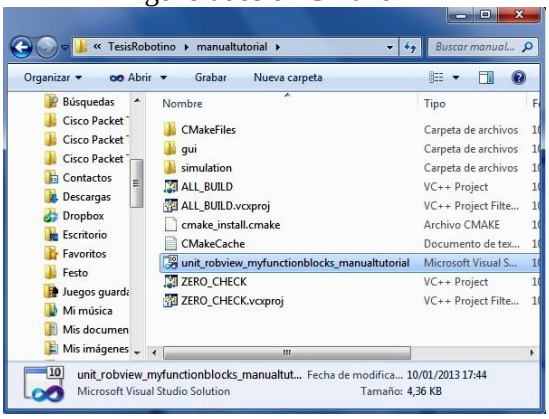


Elaborador por: Cristian Cuji.

Para comprobar que todo salió bien durante este procedimiento se revisa la dirección de la carpeta de trabajo, aquí se puede observar el proyecto de Visual Studio 2010:

C: /TesisRobotino/manualtutorial

Gráfico. 18.- Conjunto de Archivos generados en CMake



Elaborador por: Cristian Cuji.

En el gráfico anterior se observa un conjunto de archivos que fueron generados por la herramienta CMake entre ellos el archivo "unit_robotino_myfunctionblocks_manualtut

orial", al abrir este archivo se inicia el proyecto de Visual Studio 2010.

Advertencia: No abrir el archivo del Visual Studio aun, antes de empezar a programar en C++ se debe configurar.

7. Configuración

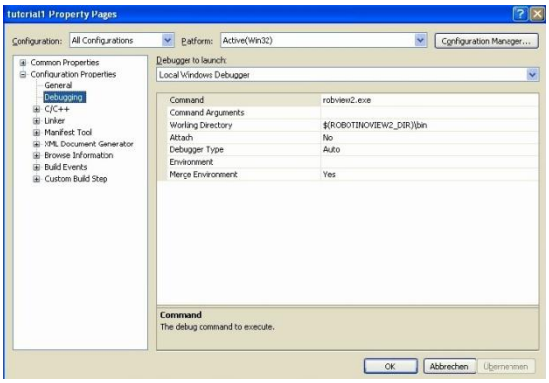
API proporciona una herramienta de configuración rápida y muy sencilla que deben ser ejecutadas antes de empezar o abrir el proyecto para programar en C++. La herramienta se llama "RUN THIS FIRST THEN START VS.cmd", solo se debe ejecutar una vez y la configuración de propiedades de página en Visual Studio cambiará. Existen dos programas ejecutables un UDP y SERVER, lo recomendable es ejecutar ambos. Los archivos se encuentran en la siguiente dirección o una similar dependiendo de la configuración del computador:

C:\Program Files(x86)\Didactic \RobotinoView2\api\examples\rec\data exchange\udp

C:\Program Files(x86)\Didactic \RobotinoView2\api\examples\rec\data exchange\server

El manual de usuario del Robotino View 2 muestra la configuración de forma manual.

Gráfico. 19.- Configuración manual de Visual Studio 2010



Elaborador por: Cristian Cuji.

Esta configuración es necesaria para el correcto funcionamiento del compilador, cuando se genera la solución del código en C++. Realizada la configuración por medio de los archivos .CMD o de forma manual ya se puede abrir y modificar el proyecto de Visual Studio.

8. Programación en Visual Studio 2010

El en este apartado busca dar las pautas necesarias para elaborar un nuevo Bloque de Funciones en RobotinoView por medio de programación en C++. Las aplicaciones, soluciones o proyectos dependen de las habilidades y capacidades del programador.

A continuación solo presento un ejemplo sencillo para familiarizarnos con el entorno de programación de Visual Studio.

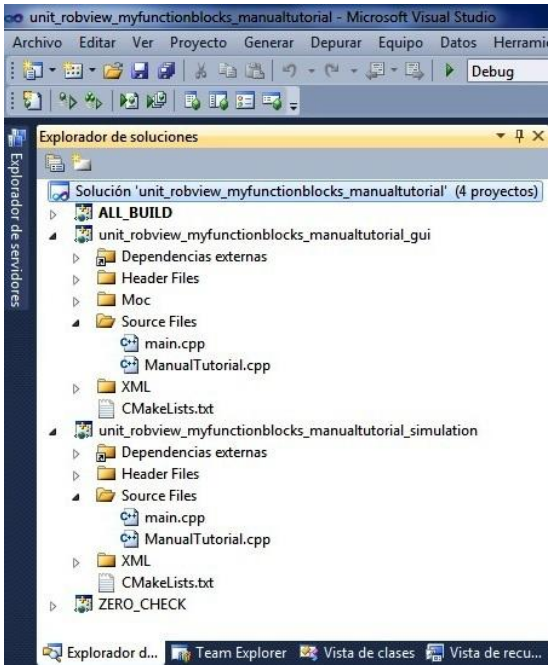
El entorno de Visual Studio 2010 nos presenta la ventana "Explorado de soluciones", donde se encuentran cuatro subproyectos, de los cuales los dos más importantes son:

"unit_robotino_myfunctionblocks_manualtutorial_gui"

"unit_robotino_myfunctionblocks_manualtutorial_simulation"

Ambos subproyectos poseen un código llamado "ManualTutorial.cpp" y "main.cpp" que contiene el código fuente que se puede editar.

Gráfico. 20.- Explorador de Soluciones de Visual Studio.



Elaborador por: Cristian Cuji.

El subproyecto manualtutorial_gui, posee dos archivos CPP. En ManualTutorial.cpp, se aprecia las entradas y salidas designadas por Function Block Manager, pero se las encuentra comentadas de la siguiente manera:

```
//del.registerInput("in0", 1);
//del.registerInput("in1", 1);
```

Para evitar errores es necesario quitar los comentarios en todas las variables, entradas y salidas.

Gráfico. 21.- ManualTutorial.cpp en GUI

Elaborador por: Cristian Cuji.

El subproyecto manualtutorial_simulation en su código fuente ManualTutorial.cpp, se aprecia un código fuente totalmente vacío, aquí es donde se va a desarrollar todo el programa en lenguaje C++. Las sentencias básicas para el desarrollo del programa son las siguientes:

```
in0 = readInput( "in0" ).toBool();
```

Lee la entrada IN0 y la identifica con un dato booleano, dependiendo de cómo se determino en el Function Block Manager, puede ser toBool, toFloat, toInt, etc.

```
out0 = in1;
```

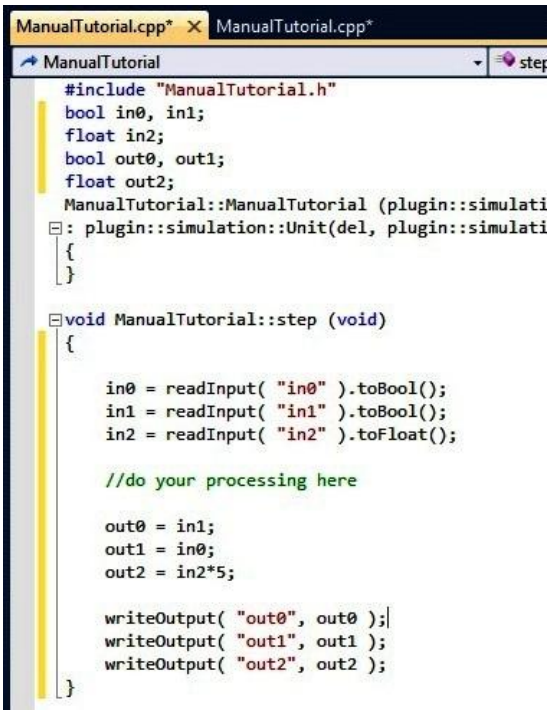
Se asigna un valor a la salida OUT0. Es importante para evitar errores que el tipo de dato sea el mismo que se determino anteriormente en la herramienta Designer Function Block.

```
writeOutput( "out0", out0 );
```

Se escribe el dato asignado en la salida del bloque de funciones.

Ejemplo de programa:

Gráfico. 22.- Programa de Ejemplo1, en C++

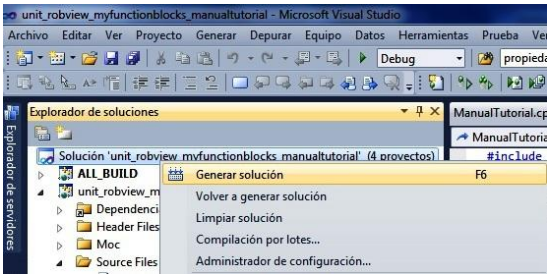


Elaborador por: Cristian Cuji.

Finalmente nunca compile el proyecto total, la compilación genera errores involuntarios propios del API.

Lo que se debe realizar es generar la solución del proyecto, esto se lo realiza dando clik derecho sobre el nombre del proyecto y clik en "Generar Solución F6".

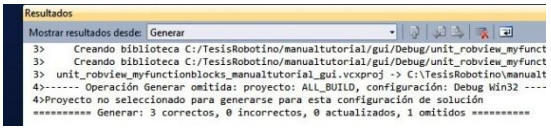
Gráfico. 23.- Generar Solución.



Elaborador por: Cristian Cuji.

De esta manera se genera una solución sin importar errores. El problema de este método es que no se puede saber si el proyecto funciona o no funciona si no hasta probar su funcionamiento en RobotinoView.

Luego de dar clik sobre Generar Solución el resultado esperado sería muy similar al que se muestra a continuación.

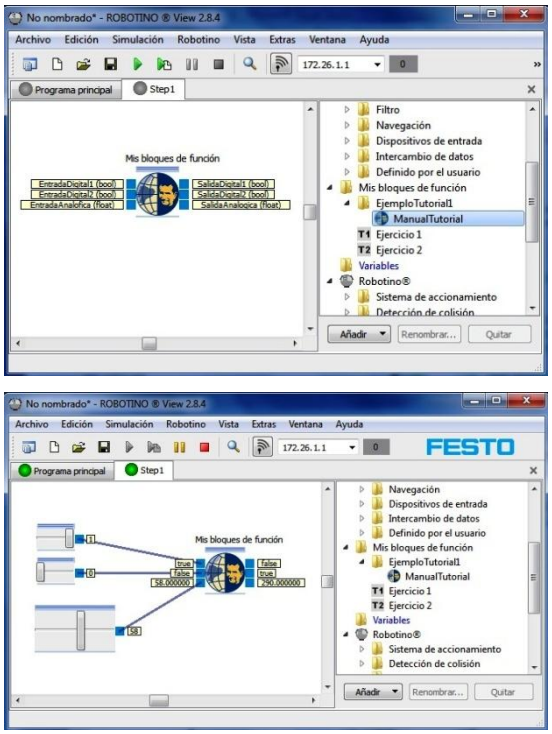


El código fuente ManualTutorial.cpp del GUI o de SIMULATION puede seguir siendo editado y generar soluciones cuantas veces sea necesario. Para realizar pruebas y verificar el funcionamiento del nuevo bloque de funciones es necesario cerrar y reiniciar RobotinoView, para actualizar los cambios realizados en Visual Studio 2010

9. Resultados finales

En RobotinoView_2.4.8 se puede apreciar un nuevo bloque de funciones y su icono que fue personalizado en Function Block Manager. Este icono posee las características programadas en las entradas / salidas. Y puede ser utilizado tanto como simulación y como un icono para el control de Robot Móvil Robotino.

Gráfico. 24.- Resultados y Simulaciones.



Elaborador por: Cristian Cuji.

CONCLUSIONES

- Un bloque de funciones es un icono personalizado al que se le puede definir entradas / salidas y variables. Además se puede modificar su comportamiento por medio de programación en lenguaje C++.
- Se deben descargar e instalar las versiones similares de RobotinoView y RobotinoView_API, para que el funcionamiento se correcto.
- No modifique el proyecto de C++ sin previamente configurar de manera manual o automático las propiedades de pagina de Visual Studio, esto se lo realiza de forma automática con los archivos UDP y SERVER de "RUN THIS FIRST THEN START VS.cmd"
- Nunca "Compile - F9" el proyecto en Visual Studio, porque se generan errores propios de RobotinoView-API. Lo correcto es "Generar la Solución – F6" del proyecto.
- Los bloques de funciones personalizados son una herramienta aun muy extensa por descubrir ya que la utilidad depende de las aplicaciones y la habilidad del programador.

REFERENCIAS

- FESTO, Alemania. *COLE_DIDACTICUM _CIA_LTDA*.2012. http://www.festo.com/net/startpage/ (último acceso: 20 de FEBRERO de 2012).
- FESTO, Bliesener Weber Karras Kling Zitzmann. *FESTO Robotino - Manual del Instructor*. Denkendorf: FESTO Didactic GmbH & Co. KG, 2007.
- RobotinoView2, FESTO. *Manual de Usuario.* Denkendorf, Germany,: © Festo Didactic GmbH & Co. KG, 2012.

ANEXOS Nº 2.- MANUAL DE USUARIO MÓDULO MANIPULADOR DE OBJETOS.

Manual de Usuario Módulo Manipulador de Objetos

Cristian Cuji C

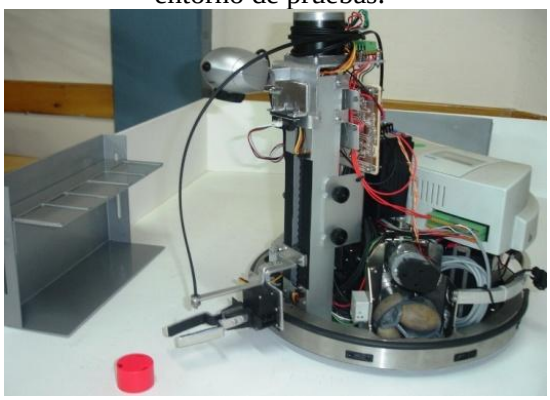
cristiancuji10@hotmail.com

Quito, marzo 2013

INTRODUCCIÓN

El módulo Manipulador de objetos es el resultado de la investigación para desarrollar un sistema mecánico y electrónico adaptable al Robot Móvil Robotino de Festo, con la capacidad de capturar objetos, elevarlos a distintas posiciones y transportarlos.

Fig. 1. Módulo manipulador de objetos y entorno de pruebas.



Elaborador por: Cristian Cuji.

El sistema consta de un elemento mecánico que se encarga de sostener los elementos o dispositivos, transmitir movimiento y sujetarse al chasis del Robotino. También posee un sistema de control electrónico encargado de interpretar las instrucciones del Robotino, generar los pulsos para el giro del motor de pasos, adaptar una señal analógica para controlar la posición de la pinza. Finalmente en el software RobotinoView v2.8.4, se implemento un driver controlador o icono que facilita la programación en lenguaje gráfico.

El presente manual tiene por objetivo dar a conocer especificaciones técnicas, acoplamiento mecánico, acoplamiento del sistema de control, software de programación, recomendaciones y posibles fallas.

1. ESPECIFICACIONES TÉCNICAS

Especificaciones Técnicas Electrónicas del Módulo	
Voltaje de alimentación del módulo.	$12V_{DC}$
Corriente Min.	50 mA
Corriente Max.	300 mA
Voltaje Motor de Pasos	$12V_{DC}$
Corriente en Motor de pasos	1,3 Amp
Entradas Digitales	4 ($24V_{DC}$)
Entradas Analógicas	1 (10V)
Motor de Pasos	1 (6 cables)
Servomotor	1 (3 cables)
Sensor Fin de Carrera	2 (3 cables)

Elaborador por: Cristian Cuji.

Especificaciones Técnicas Mecánicas	
Altura del módulo	30 cm
Rango de funcionamiento	15 cm
Material	Aluminio
Tornillo sin fin	Acero de transmisión
Diámetro tornillo sin fin	14 mm.
Transferencia de movimiento	Engranaje circular
Mecanismo Gripper	Plástico

Elaborador por: Cristian Cuji.

Especificaciones Técnicas Software.	
Software	Robotino View v2.8.4
	Cmake - gui 2.8.4
	Visual Studio 2010
	Function Block Manager

Elaborador por: Cristian Cuji.

Salidas Robotino	
Entrada Digital 1	24VDC
Entrada Digital 2	
Entrada Digital 3	
Entrada Digital 4	
GND	0V (Referencia)
Salida Analógica	0,46V - 2,1V

Elaborador por: Cristian Cuji.

Borneras	
Bobina A	12 VDC Motor de pasos.
Bobina B	
Bobina C	
Bobina D	
Bobina Común	0VDC
Fin de Carrera 1	Contacto N.A.
Fin de Carrera 2	Contacto N.A.

Elaborador por: Cristian Cuji.

Indicaciones:

1.1. Verificar datos técnicos, sobretodo conexiones y borneras bien sujetas.

1.2. Verificar voltajes de la batería de $12V_{DC}$, el rango apropiado de voltaje debe ser entre ($12,9V_{DC}$ y no menor a $10,0V_{DC}$).

1.3. Identificar todos los tornillos, ranuras y cables de conexión necesarios.

2. ACOPLAMIENTO MECÁNICO

El acoplamiento del módulo al chasis del Robotino es sencillo, la estructura metálica del módulo consta de ranuras donde se pueden sujetar los distintos elementos necesarios.

En la base se encuentran ranuras longitudinales para calibrar la posición del sistema mecánico, es importarte colocar el módulo de tal forma que durante el descenso no choque el gripper con el chasis del Robotino. Esto puede ocurrir si las ranuras longitudinales de la base del módulo no se encuentran en el centro y lo más cercano al extremo.

Debajo de la tarjeta de control existe un espacio con dos perforaciones de tornillo

para adaptar un sensor óptico de fibra. La cámara web también posee una ranura de tornillo.

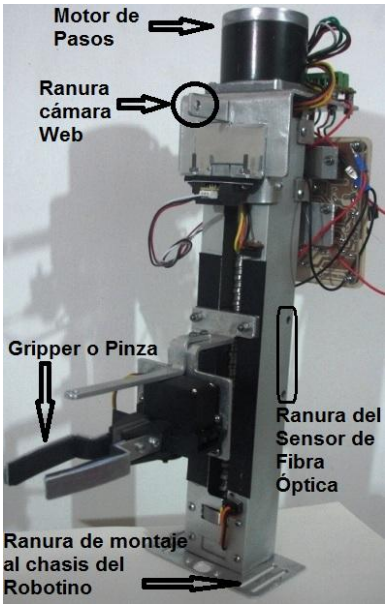
Mecánicamente el sistema es adaptable al Robot Móvil Robotino sin mayores detalles técnicos por el contrario es muy sencillo ya que solo depende de ajustar tornillos y tuercas entre el modulo y el chasis del Robotino.

Principalmente se debe sujetar bien la base del módulo al chasis del Robotino esto se lo consigue con 4 tornillos y tuercas que calzan perfectamente uno con el otro. El sensor de fibra óptica y la cámara web posee ranuras claramente identificadas en la figura anterior.

El gripper o pinza es un elemento que consta de un mecanismo plástico y un servomotor, al cual se ha adaptado una extensión de aluminio, estas pueden ser cambiadas o ajustadas de acuerdo a la necesidad.

La tarjeta de control también posee un espacio en la estructura mecánica donde se puede sujetar por medio de ocho tornillos.

Fig. 2. Elementos mecánicos y ranuras para sujetar el modulo al chasis del Robotino



Elaborador por: Cristian Cuji.

Indicaciones:

2.1. Colocar el módulo sobre el chasis del Robotino.

2.2. Calibrar o buscar la posición adecuada donde el chasis del Robotino no estorbe en el movimiento de descenso del gripper, esta calibración se la realiza visualmente.

2.3. En las ranuras de la base se deben colocar dos tornillos uno en cada ranura longitudinal, dos tornillos más serán ubicados en dos agujeros que cuadran perfectamente entre el módulo y el chasis del Robotino, si el módulo se encuentra bien calibrado.

2.4. Ajustar bien los tornillos de la base.

2.5. Una vez acoplado el módulo se puede colocar los elementos adicionales

como la cámara web, el sensor óptico y la batería.

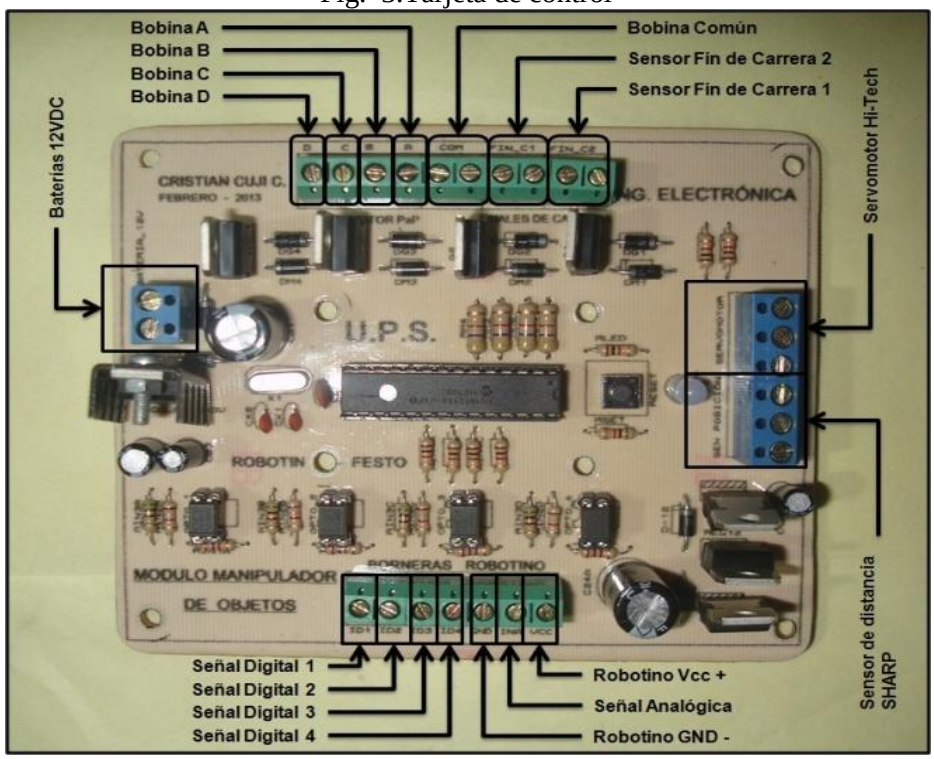
2.6. Ajustar no con fuerza pero si con firmeza los tornillos.

3. ACOPLAMIENTO SISTEMA DE CONTROL

El sistema electrónico es una tarjeta de control y adquisición de datos basada en un microcontrolador. Sus funciones son:

- Control del motor de pasos y servomotor.
- Adaptar señal analógica de posición.
- Interpretar señales provenientes del Robotino.
- Leer los sensores fin de carrera.

Fig. 3.Tarjeta de control



Elaborador por: Cristian Cuji.

En el anterior gráfico se visualiza todas las borneras de la tarjeta de control.

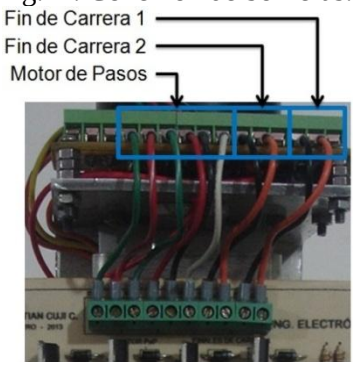
De color azul se observan: La Batería, Servomotor, Sensor de distancia.

De color verde en la parte superior se encuentran las borneras de todos los periféricos: Motor de pasos y sensores fin de carrera. En la parte inferior se encuentran las borneras de conexión hacia el Robotino: 4 señales digitales, 1 señal analógica y la fuente de $24V_{DC}$.

Las bobinas A, B, C, D y Común pertenecen al motor de pasos. Los sensores finales de

carrera 1 y 2 son seguridades que actúan como contactos normalmente abiertos, la conexión se realiza de forma directa a la tarjeta.

Fig. 4. Conexión de borneras.



Elaborador por: Cristian Cuji.

Como se aprecia en el gráfico anterior, previamente en el sistema mecánico existen un conjunto de borneras o terminales de todos los elementos eléctricos y electrónicos: Motor PaP, Finales de carrera, servomotor. Estas borneras están organizadas de tal forma que se conectan paralelamente entre si desde la tarjeta de control al sistema mecánico.

Hacia las borneras llegan absolutamente todos los cables de conexión, y en la tarjeta de control no necesariamente se usan todas las señales, sino las que son necesariamente útiles, esto se puede apreciar en la Fig. 3.Tarjeta de control

Las borneras azules son por un lado la fuente necesaria para el funcionamiento del motor de pasos y toda la tarjeta, aquí se conecta una batería de $12V_{DC}$ - 2Amp. Y el conjunto de 6 borneras azules están divididas entre el sensor de distancia y el servomotor.

Servomotor	Señal
	Vcc (5V)
	GND1
Sensor Posición	Señal
	Vcc (5V)
	GND2
Batería 12V	VCC (12V)
	GND

Elaborador por: Cristian Cuji.

Para el Robotino se conecta el conjunto de 7 borneras, las cuatro primeras son salidas digitales de $24V_{DC}$, la quinta es el nivel de referencia o GND, la sexta bornera es una señal análoga que ingresa con la información de posición y finalmente la séptima bornera es una fuente de $24V_{DC}$ para el funcionamiento del sensor.

Es decir para el funcionamiento del sistema manipulador de objetos se necesita conectar desde el Robotino:

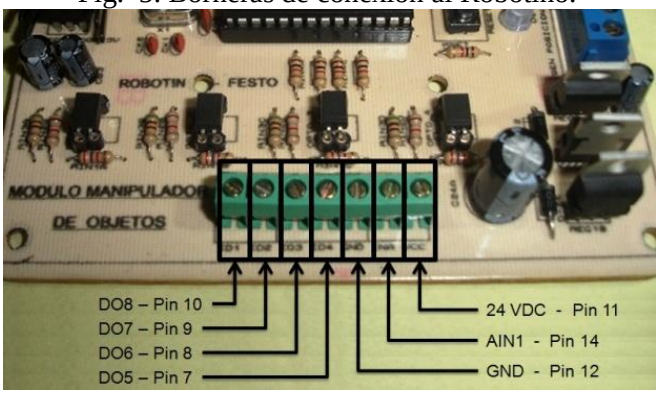
- Una Fuente de $24V_{DC}$
- Una entrada analógica.
- Cuatro salidas digitales.

Como ejemplo se recomienda la siguiente conexión:

DO8	Gripper
DO7	Secuencia 1 - Subir
DO6	Secuencia 2 - Bajar
DO5	ON / OFF

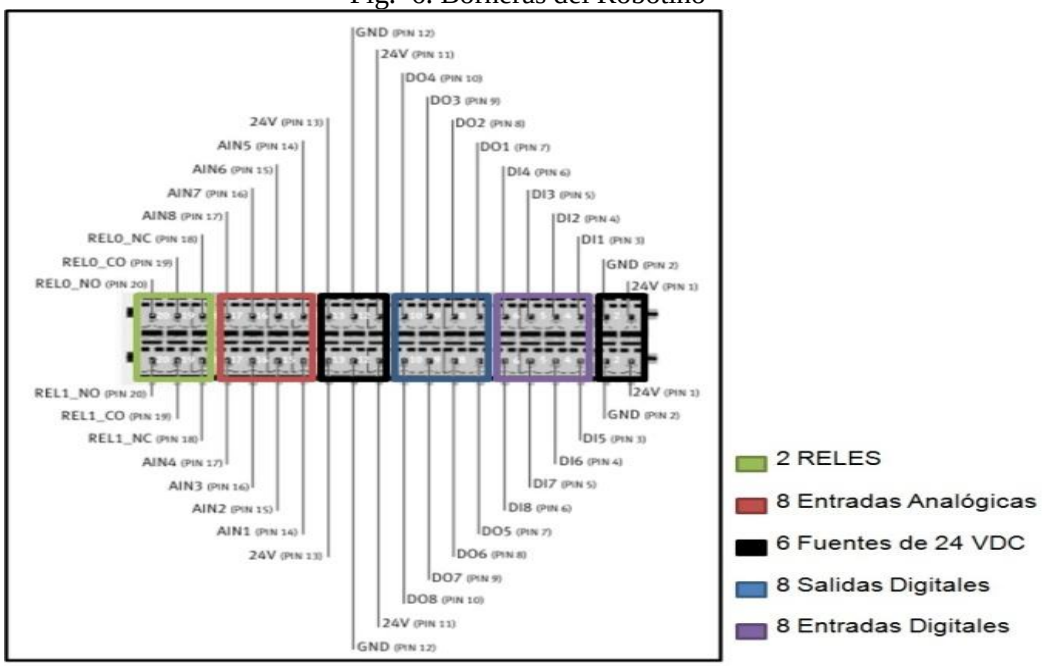
Para un mayor detalle de la conexión física a continuación se muestra la distribución de las borneras del Robotino.

Fig. 5. Borneras de conexión al Robotino.



Elaborador por: Cristian Cuji.

Fig. 6. Borneras del Robotino

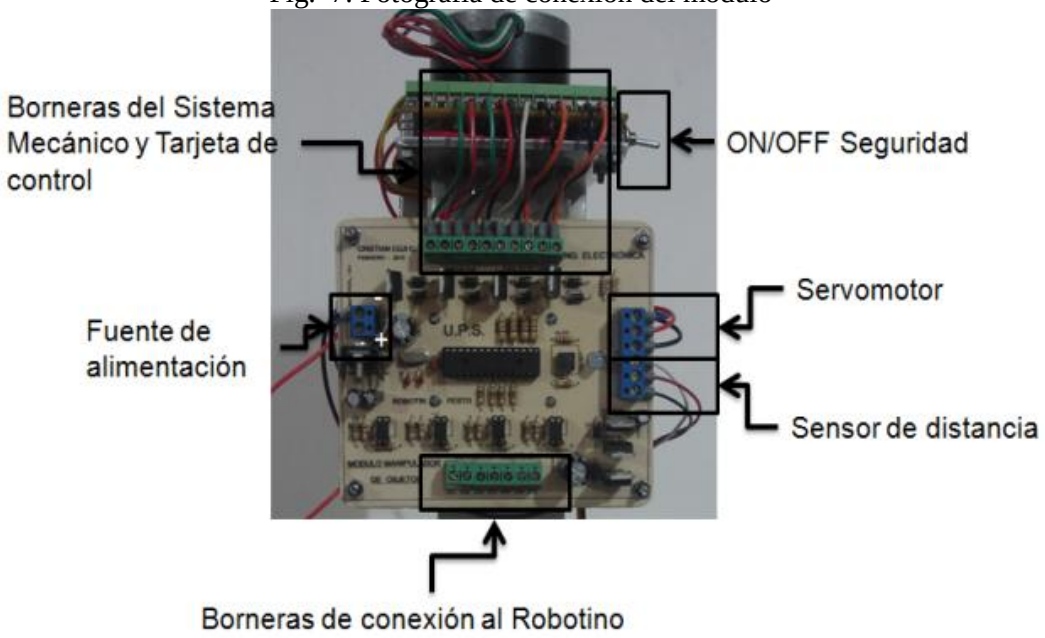


Elaborador por: Cristian Cuji.

El sistema incluye el uso del Rele_N.O. Pin 19 y 20, que activa y desactiva todo el sistema electrónico, la principal razón de utilizar un sistema de conexión y desconexión en la batería es para ahorrar la energía.

Las conexione se realizan en la mayoría de casos con cable flexible 20 AWG.

Fig. 7. Fotografía de conexión del módulo



Elaborador por: Cristian Cuji.

Indicaciones

3.1. Colocar la tarjeta de control junto a las borneras del sistema mecánico.

3.2. De acuerdo a las indicaciones y especificaciones de las borneras se debe conectar y ajustar desde el sistema mecánico a la tarjeta de control el motor de pasos, sensores y servomotor.

3.3. En perfecto orden siguiendo las especificaciones de conexión se debe conectar las borneras de la tarjeta de control hacia el Robotino.

3.4. Finalmente se debe conectar la batería.

4. DRIVER DE CONTROL

El driver de control es un icono programado en lenguaje C++ y desarrollado para facilitar la programación en Robotino View v2.8.4 que es un lenguaje gráfico de programación propio del Robot Móvil Robotino.

Como ya se describió en las especificaciones técnicas, son necesarios los siguientes requerimientos, para un buen funcionamiento del sistema:

- Robotino View v2.8.4
- Robotino View API v2.8.4
- OPEN Robotino.
- Cmake - gui v2.8.4
- Function Block Manager.
- Visual Studio 2010.
- Windows 7 o Xp.
- Procesador igual o superior a Core Duo.

El presente apartado no es un manual de desarrollo del driver, se supone que ya existe el icono y se encuentra totalmente funcional. Caso contrario es recomendable revisar el tutorial **"BLOQUE DE FUNCIONES" PARA ROBOTINO VIEW 2.8.4,** que se encuentra como anexo en el proyecto de tesis. El Icono se encuentra en: *Barra de Herramientas/ Mis bloques de funciones/ ##dirección##:*

Características de Posicionamiento

Al modulo manipulador de objetos se le ha dividido en 12 posibles posiciones, en un rango de libertad de 15cm. La tabla siguiente muestra la distribución de las posiciones y su equivalencia en centímetros.

Posición	Ubicación	Descripción
-1	0,1 cm	Punto final Inferior
0	0,5 cm	Capturar del piso 1
1	1 cm	Capturar del piso 2
2	3,5 cm	Posición elevada 1
3	5 cm	Posición elevada 2
4	6,5 cm	Posición elevada 3
5	8 cm	Posición elevada 4
6	9,5 cm	Posición elevada 5
7	11 cm	Posición elevada 6
8	12,5 cm	Posición elevada 7
9	14 cm	Posición elevada 8
10	14,5 cm	Punto final Superior

Elaborador por: Cristian Cuji.

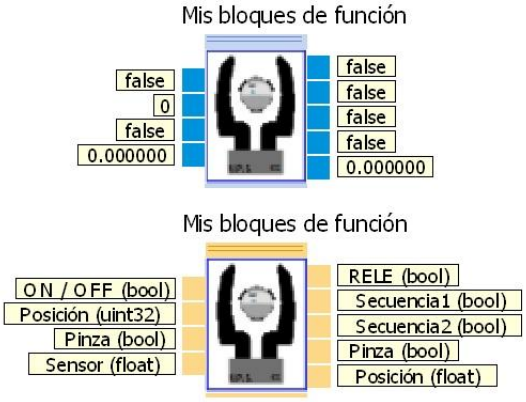
La posición -1 es una posición con la que prácticamente se raspa el piso con el gripper, por esta razón es recomendable utilizar dos posiciones de aproximación: la posición 1 y 2. Desde la posición 3 al 9 son posiciones elevadas y la posición 10 es una posición de seguridad en el punto final superior.

Características del Icono

La programación del icono es totalmente lógica, las características y detalles se asignan en la configuración previa. Los iconos desarrollados como bloques de funciones tienen todas las características de un icono propio del software.

En el siguiente gráfico se puede visualizar dos características principales: Descripción de conector y Valores de conector.

Fig. 8. Driver controlador del módulo manipulador de objetos



Elaborador por: Cristian Cuji.

El driver posee cuatro entradas y cinco salidas, internamente alrededor de 5 variables.

Las características como tipo de variable, nombre, y conexiones físicas de los conectores de entradas y salidas del driver se visualizan a continuación en el siguiente cuadro.

Entradas				
Nombre	**Tipo de Variable**	**Conector**		
ON/OFF	Bool	Interno		
Posición	Int	Interno		
Pinza	Bool	Entrada Digital 6	DIN6	Pin 4
Sensor	Float	Entrada Analógica 1	IAN1	Pin14

Salidas				
Nombre	**Tipo de Variable**	**Conector**		
Rele	Bool	Relé 1	Ain 1	Pin19 - Pin20
Secuencia1	Bool	Salida digital 8	DO8	Pin6
Secuencia2	Bool	Salida digital 7	DO7	Pin7
Pinza	Bool	Salida digital 6	DO6	Pin8
Posición	Int	Salida digital 5	DO5	Pin9

Elaborador por: Cristian Cuji.

ON/OFF:

Enciende o apaga la tarjeta de control.

Posición: Es un valor del 0 al 10 que indica la posición en la que se desea ubicar al gripper.

Pinza: 0 Abrir pinza / 1 Cerrar pinza, siempre que la tarjeta se encuentre encendida.

Sensor: es el valor que se recibe desde el sensor de posición.

Rele: Se activa o desactiva como un switch para encender o apagar la tarjeta de control.

Secuencia 1: Inicia la secuencia de giro en sentido horario es decir permite ascender al gripper.

Secuencia 2: Inicia la secuencia de giro en sentido anti horario es decir permite descender al gripper.

Pinza: Tiene el control de apertura o cierre del gripper o pinza.

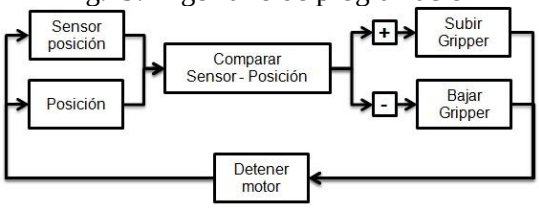
Posición: Indica la posición en la que se encuentra el gripper.

Descripción de funcionamientos

El sensor lee la posición del gripper y compara con la posición deseada, el sistema electrónico ascenderá o descenderá según sea necesario hasta que la comparación sea cero.

Con ON/OFF se puede activar la tarjeta electrónica y solo en ese momento se puede abrir o cerrar el gripper.

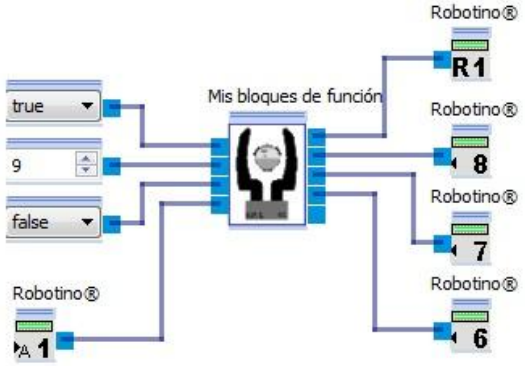
Fig. 9. Algoritmo de programación



Elaborador por: Cristian Cuji.

Finalmente en el siguiente gráfico se observa un ejemplo de cómo utilizar o probar el nuevo bloque de funciones.

Fig. 10. Ejemplo de conexión



Elaborador por: Cristian Cuji.

Indicaciones:

4.1. Crear el nuevo bloque de funciones con la ayuda de la herramienta Function Block Manager.(Revisar Manual de bloques de funciones.)

4.2. Traducir el código generado en el paso anterior con la ayuda de CMake - gui en un proyecto de Visual Studio 2010.

4.3. Programar en Visual Studio.

4.4. Terminado el proceso de programación se debe agregar el bloque de funciones al software Robotino View v2.8.4. Esto se consigue con el software Function Block Manager.

5. POSIBLES FALLAS

Batería descargada: Es la falla más concurrente, produce atrancamientos mecánicos y es producida por un extenso tiempo de uso del modulo aproximadamente 1 hora seguida.

Ruidos electromagnéticos: La tarjeta tiene como base un microcontrolador de familia MickroChip, este es un dispositivo que puede ser fácilmente victima de ruido electromagnético. Aunque el circuito tiene filtros, mallas de tierra y protecciones, grandes interferencias como la antena de un teléfono celular si puede alterar el comportamiento de la tarjeta de control.

Atascamientos mecánicos: Se producen cuando existen pequeños granos de polvo o basuras que puedan atrancar el tornillo sin fin, lo recomendable es hacer funcionar al sistema en un lugar limpio.

Histéresis en determinadas posiciones: Existen puntos donde por características del sensor los valores de medición en subida o en

bajada se ven alterados produciendo en una misma posición dos diferente valores. Esto ocurre en las posiciones intermedias más alejadas del sensor: Posición 2, 3 y 4. La posición 1 esta corregida por un factor mecánico y siempre es precisa.

6. CONCLUSIONES Y RECOMENDACIONES

No se preocupe al apagar totalmente la tarjeta de control, porque el modulo manipulador posee un atrancamiento mecánico en el tornillo sin fin y en el gripper lo cual evita que el gripper se resbale o descienda y también que la pinza suelte el objeto transportado.

El modulo es mucho más eficiente en tareas de descenso con cargas y transporte de objetos. El ascenso de objetos produce un mayor desgaste de la batería.

Se debe tener mucho cuidado con el desplazamiento del Robotino cuando este montado el módulo, un choque puede romper y afectar seriamente no solo al modulo manipulador sino también al Robotino.

7. BIBLIOGRAFÍA

C, Cristian Cuji. *"Bloque de Funciones" para Robotino View v2.8.4.* Quito: Trabajo de Tesis, 2012.

RobotinoView2, FESTO. *Manual de Usuario.* Denkendorf, Germany,: © Festo Didactic GmbH & Co. KG, 2012.

ANEXOS Nº 3.- SENSOR GP2D120 DATA SHEET.

GP2D120

Optoelectronic Device

SHARP

FEATURES

- Analog output
- Effective range: 4 to 30 cm
- Typical response time: 39 ms
- Typical start up delay: 44 ms
- Average Current Consumption: 33 mA

DESCRIPTION

The GP2D120 is a distance measuring sensor with integrated signal processing and analog voltage output.

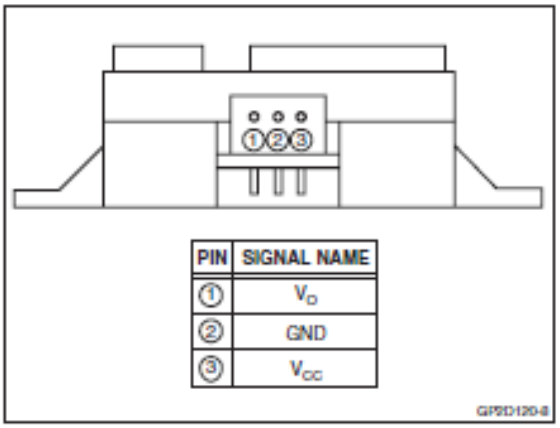


Figure 1. Pinout

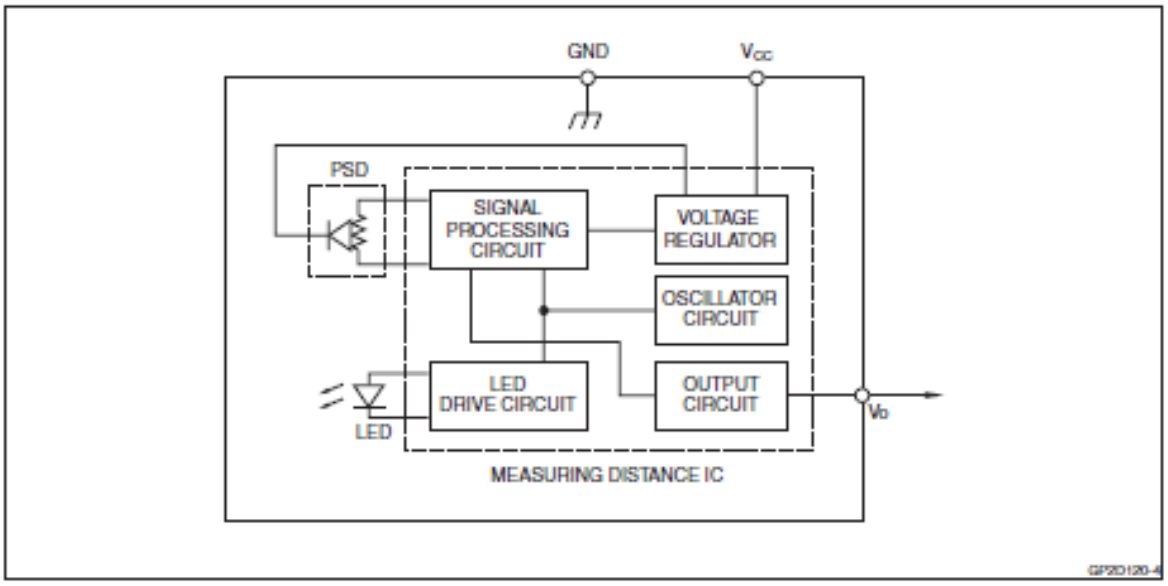


Figure 2. Block Diagram

ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings

Ta = 25°C, V_{CC} = 5 VDC

PARAMETER	SYMBOL	RATING	UNIT
Supply Voltage	V_{CC}	-0.3 to +7	V
Output Terminal Voltage	V_O	-0.3 to (V_{CC} +0.3)	V
Operating Temperature	Topr	-10 to +60	°C
Storage Temperature	Tstg	-40 to +70	°C

Operating Supply Voltage

PARAMETER	SYMBOL	RATING	UNIT
Operating Supply Voltage	V_{CC}	4.5 to 5.5	V

Electro-optical Characteristics

Ta = 25°C, V_{CC} = 5 VDC

PARAMETER	SYMBOL	CONDITIONS	MIN.	TYP.	MAX.	UNIT	NOTES
Measuring Distance Range	ΔL		4	—	30	cm	1, 2
Output Terminal Voltage	V_O	L = 30 cm	0.25	0.4	0.55	V	1, 2
Output Voltage Difference	ΔV_O	Output change at ΔL (30 cm – 4 cm)	1.95	2.25	2.55	V	1, 2
Average Supply Current	I_{CC}	L = 30 cm	—	33	50	mA	1, 2

NOTES:
1. Measurements made with Kodak R-27 Gray Card, using the white side, (90% reflectivity).
2. L = Distance to reflective object.

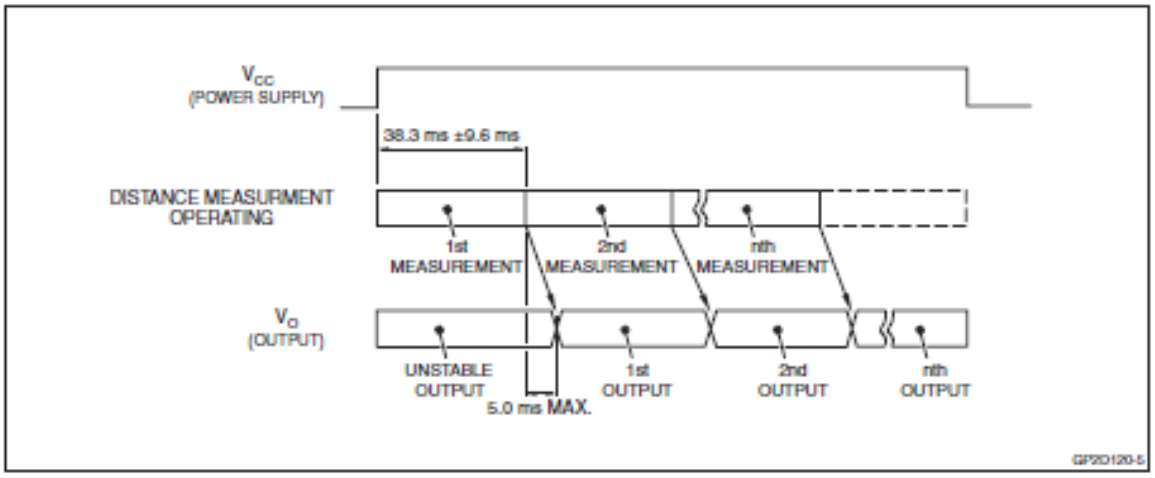


Figure 3. Timing Diagram

REALIABILITY

The reliability of requirements of this device are listed in Table 1.

Table 1. Reliability

TEST ITEMS	TEST CONDITIONS	FAILURE JUDGEMENT CRITERIA	SAMPLES (n), DEFECTIVE (C)	NOTES
Temperature Cycling	One cycle -40°C (30 min.) to +70°C in 30 minutes, repeated 25 times	Initial × 0.8 > V_O V_O > Initial × 1.2	n = 11, C = 0	1
High Temperature and High Humidity Storage	+40°C, 90% RH, 500h		n = 11, C = 0	1
High Temperature Storage	+70°C, 500h		n = 11, C = 0	1
Low Temperature Storage	-40°C, 500h		n = 11, C = 0	1
Operational Life (High Temperature)	+60°C, V_{CC} = 5 V, 500h		n = 11, C = 0	1
Mechanical Shock	100 m/s², 6.0 ms 3 times/±X, ±Y, ±Z direction		n = 6, C = 0	1
Variable Frequency Vibration	10-to-55-to-10 Hz in 1 minute Amplitude: 1.5 mm 2h in each X, Y, Z direction		n = 6, C = 0	1

NOTES:
1. Test conditions are according to Electro-optical Characteristics, shown on page 2.
2. At completion of the test, allow device to remain at nominal room temperature and humidity (non-condensing) for two hours.
3. Confidence level: 90%, Lot Tolerance Percent Defect (LTPD): 20%/40%.

MANUFACTURER'S INSPECTION

Inspection Lot

Inspection shall be carried out per each delivery lot.

Inspection Method

A single sampling plan, normal inspection level II based on ISO 2859 shall be adopted.

Table 2. Quality Level

DEFECT	INSPECTION ITEM/TEST METHOD	AQL (%)
Major Defect	Electro-optical characteristics defect	0.4
Minor Defect	Defect on appearance and dimension (split, chip, scratch, stain)*	1.0

NOTE: *Any one of these that affects the Electro-optical Characteristics shall be considered a defect.

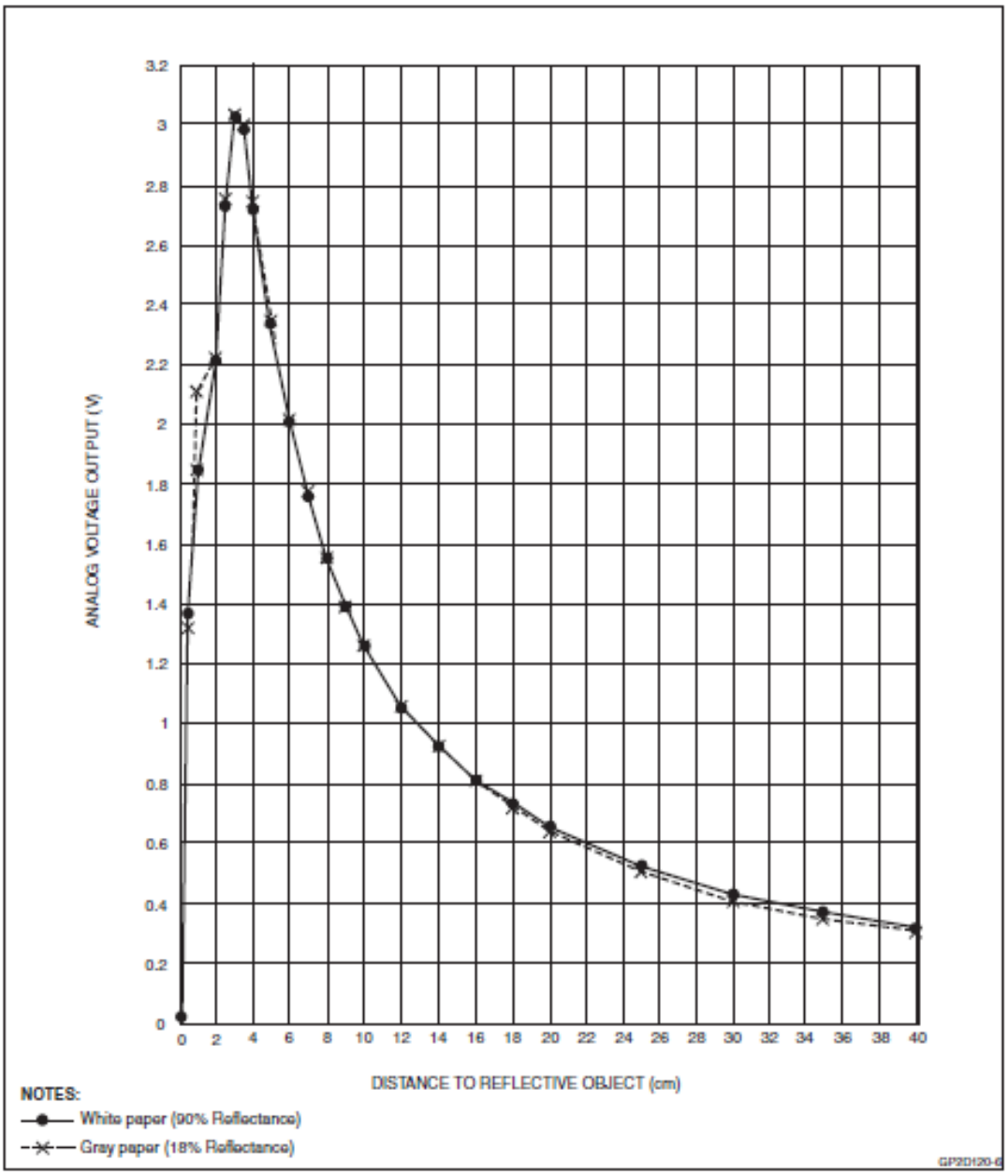


Figure 4. GP2D120 Example of Output Distance Characteristics

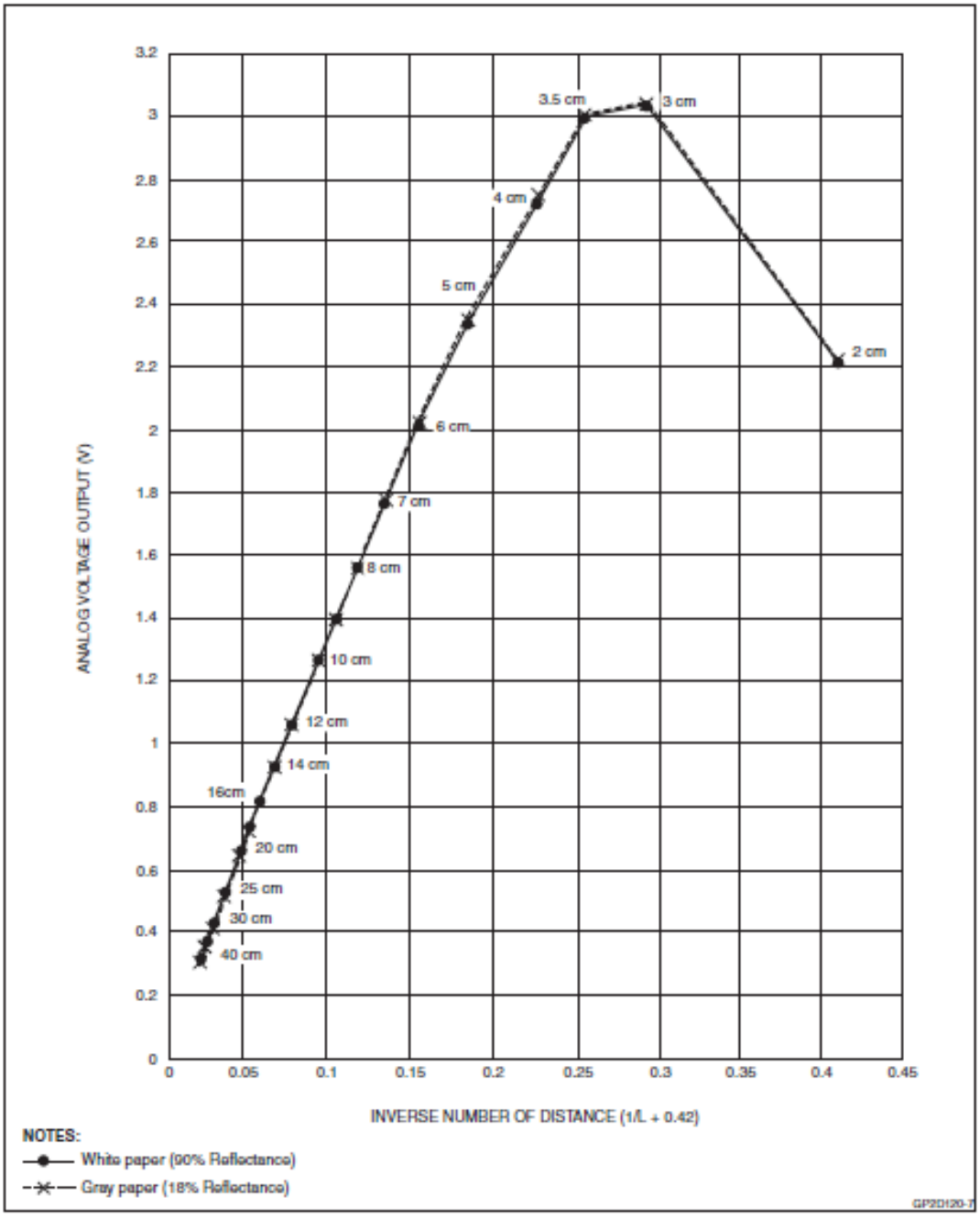
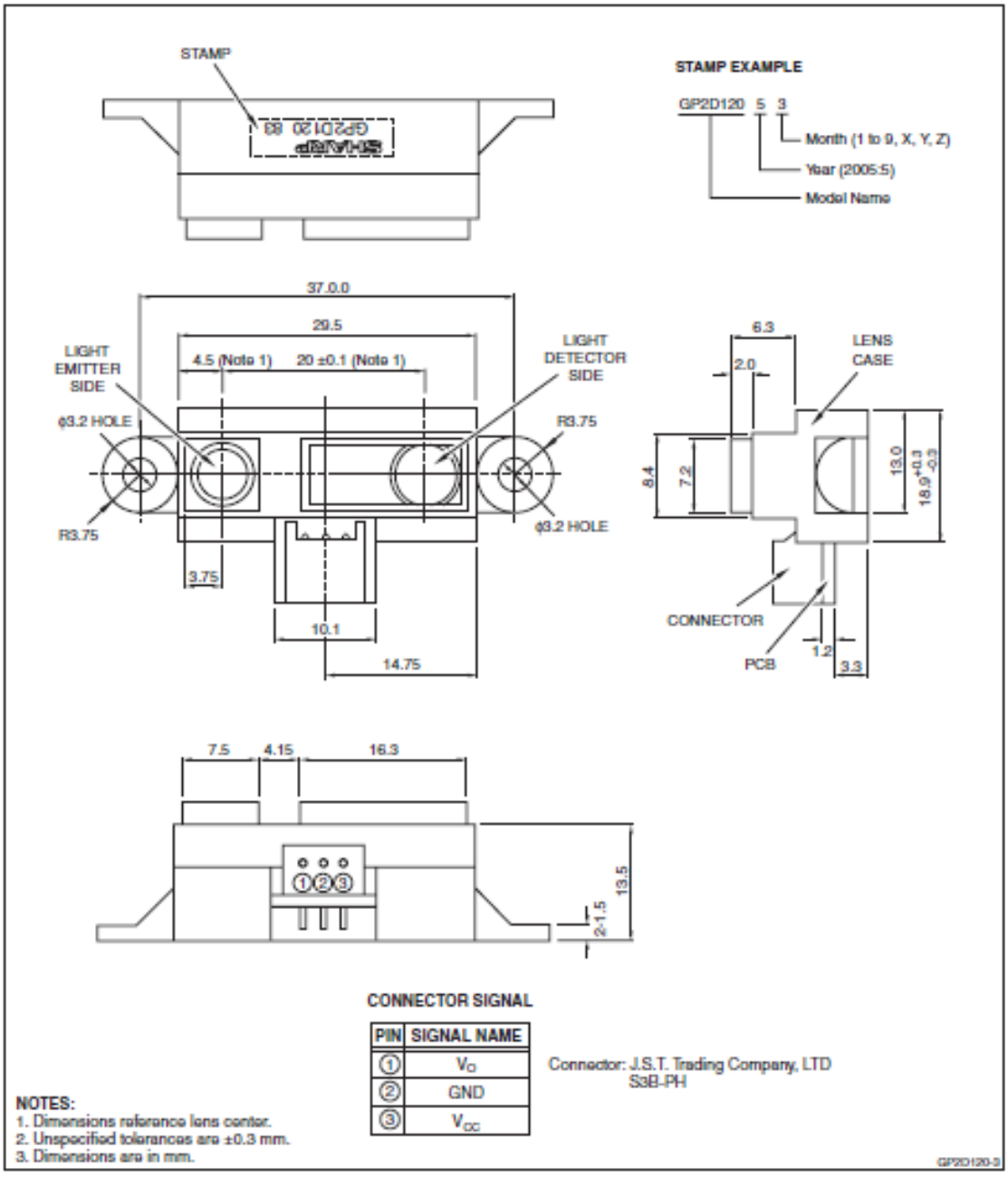


Figure 5. GP2D120 Example of Output Characteristics with Inverse Number of Distance

PACKAGE SPECIFICATIONS



CONNECTOR SIGNAL

PIN	SIGNAL NAME
①	V_O
②	GND
③	V_{CC}

Connector: J.S.T. Trading Company, LTD
S3B-PH

NOTES:
1. Dimensions reference lens center.
2. Unspecified tolerances are ±0.3 mm.
3. Dimensions are in mm.

GP2D120-3