UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA: INGENIERÍA ELECTRÓNICA

Tesis previa a la obtención del título de: INGENIERO ELECTRÓNICO E INGENIERA ELECTRÓNICA

TEMA:
DISEÑO E IMPLEMENTACIÓN DE ALGORITMOS DE CINEMÁTICA DIRECTA E INVERSA UTILIZANDO FPGA'S PARA CONTROLAR EL MANIPULADOR ROBÓTICO SCARA (SELECTIVE COMPLIANT ARTICULATED ROBOT ARM) PERTENECIENTE A LA UNIVERSIDAD POLITÉCNICA SALESIANA CAMPUS SUR.

AUTOR:
ALEJANDRA MARIANELA ORTIZ CRUZ
EDISON FERNANDO PONCE PACHECO

DIRECTOR:
CARLOS GERMÁN PILLAJO ANGOS

Quito, junio del 2014

DECLARATORIA DE RESPONSABILIDAD Y AUTORIZACIÓN DE USO DEL TRABAJO DE TITULACIÓN

Nosotros, autorizamos a la Universidad Politécnica Salesiana la publicación total o parcial de este trabajo de titulación y su reproducción sin fines de lucro.

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

_____	_____
Alejandra Marianela Ortiz Cruz	Edison Fernando Ponce Pacheco
CI: 0502472566	CI: 1717202616

DEDICATORIA

A Dios.

Por habernos permitido llegar a esta instancia de nuestras vidas, por darnos la inteligencia y sabiduría para realizar este proyecto, y por habernos dado salud para lograr nuestros objetivos, además de su infinita bondad y amor.

A nuestros Padres.

Por habernos apoyado en todo momento, por su paciencia, sus consejos y valores, pero más que nada, la motivación constante que nos brindaron para ser personas de bien.

Alejandra Marianela Ortiz Cruz &
Edison Fernando Ponce Pacheco

AGRADECIMIENTO

Agradecemos a nuestros profesores, quienes aportaron sus conocimientos con gran voluntad para construir nuestras vidas profesionales, agradecemos a nuestros compañeros de aula, con quienes compartimos experiencias que ayudaron y ayudaran a formar ese vínculo de amistad que permanecerá arraigado a nuestros corazones.

Alejandra Marianela Ortiz Cruz &
Edison Fernando Ponce Pacheco

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

ÍNDICE DE ANEXOS

RESUMEN

En este proyecto de titulación se ha diseñado e implementado un algoritmo cinemático directo e inverso que permite la manipulación de un brazo robótico RHINO Scara a través de la interacción entre una HMI (Interfaz Maquina Humano) por diseñar y un hardware ya existente. La interfaz gráfica del software se la realizó mediante LabVIEW con una comunicación serial entre la computadora y el hardware, además se programó dicho algoritmo en una tarjeta Spartan 3E previamente programada con un código VHDL en el programa Project Navigator proporcionado por Xilinx. Se debe resaltar que este proyecto tiene fines académicos para la Universidad Politécnica Salesiana del Campus Sur, debido a que el brazo robótico servirá como una herramienta del laboratorio de robótica y materias afines que diariamente los estudiantes utilizan en las practicas, desarrollando así una visión general del funcionamiento del código implementado como las varias aplicaciones que se pueden tener.

ABSTRACT

This draft grade was designed and implemented an algorithm for manipulating a RHINO Scara robotic arm through the interaction between a Human Machine Interface for designing and an existing hardware. The graphic interface of the software was done through LabVIEW with the serial communication between the computer and the hardware, also the algorithm was implemented on Spartan 3E target pre-programmed with VHDL code in the program provided by Xilinx Project Navigator. It should be noted that this project is for academic purposes to Universidad Politécnica Salesiana Campus Sur, because the robotic arm will serve as a robotic laboratory tool that students use in daily practices, developing an overview of the implemented code works as several applications that can be had.

INTRODUCCIÓN

El estudio de robótica se ha vuelto muy importante en la industria ya que es indispensable para procesos en donde se requiere mayor esfuerzo, velocidad y alto grado de precisión y exactitud, por lo que se ve la necesidad de la implementación de brazos robóticos para realizar diversos procesos.

En este proyecto se actualizó el hardware y software de un brazo robótico RHINO Scara con cinco grados de libertad, los cuales actúan a través de cinco motores acoplados en diferente posición para realizar los movimientos de rotación y desplazamiento, estos motores están constituidos por encoders los cuales permiten estimar las posiciones de las diferentes partes del robot Scara.

El robot Scara está constituido por cinco partes: brazo, hombro, codo, muñeca, mano los cuales imitan el movimiento de un brazo humano, para el control de los motores se acopló una tarjeta FPGA Spartan 3E de Xilinx, la cual es un conjunto de arreglos que produce señales digitales para el movimiento de los motores y la captación de la señal digital de los encoders.

Para el control del robot Scara se utilizó una interfaz gráfica con comunicación serial entre el software LabVIEW y la tarjeta Spartan 3E, en la cual se presenta los datos referentes para: movimientos de izquierda y derecha y ángulos de giro para los motores que controlan cada parte del robot. El software LabVIEW envía la dirección y ángulo de giro encapsulada en una variable tipo string por el puerto serial hacia la tarjeta Spartan 3E, indicando al brazo robótico los movimientos que éste debe realizar, la tarjeta Spartan 3E recibe los datos enviados por el software LabVIEW y los desencapsula para identificar y activar los movimientos de los motores del robot Scara.

Quedando la investigación dividida de la siguiente forma:

Capítulo 1: se presenta los objetivos, justificación y alcance los cuales llevaron a la realización de este proyecto de titulación.

Capítulo 2: se especifica el marco teórico utilizado para control y comunicación del robot Scara, así como los fundamentos matemáticos para el cálculo de rotación y funcionamiento de los motores.

Capítulo 3: se describe el hardware y software utilizados en el diseño e implementación del robot Scara.

Capítulo 4: se detalla el costo total del desarrollo del proyecto de titulación.

Además, se presenta las conclusiones y recomendaciones que surgieron en la realización y culminación del proyecto, también se presenta anexos en los cuales detallan características importantes del proyecto, tanto de información técnica como de programación.

CAPÍTULO 1
PLANTEAMIENTO DEL PROBLEMA

1.1. Problema a resolver

La historia de la Robótica ha estado unida a la construcción de "artefactos", muchas veces por obra de genios autodidactas que trataban de materializar el deseo humano de crear seres semejantes a nosotros que faciliten el trabajo.

En la actualidad muchas fábricas utilizan brazos robóticos para el ensamblaje de sus productos, ya que la eficiencia en producción que se logra es muy alta.

El campo de la robótica es desarrollado en Universidades Politécnicas, muchas de las cuales cuentan con laboratorios equipados con robots de varios tipos, los cuales sirven para enseñar al alumno de una manera didáctica su funcionamiento.

La Universidad Politécnica Salesiana Campus Sur, en su laboratorio de robótica, posee algunos brazos robóticos, entre estos el Robot RHINO Scara, el cual consta de controladores para su manejo. Con el avance de la tecnología, estas tarjetas de control han quedado obsoletas y en inactividad.

Frente a esta situación se detectó la necesidad de actualizar el hardware y software de control del manipulador robótico para obtener los resultados deseados.

El proyecto consiste en utilizar una tarjeta FPGA Spartan 3E para la implementación de algoritmos cinemáticos, brindando gran velocidad de transmisión de datos, para el movimiento del manipulador robótico. En la creación de los algoritmos se utilizará el código fuente VHDL, además se implementará una interfaz amigable que será utilizada por los estudiantes del Campus Sur.

1.2. Objetivos

1.2.1. Objetivo general.

Diseñar e Implementar algoritmos de cinemática directa e inversa utilizando FPGA’s para controlar el manipulador robótico SCARA (SELECTIVE COMPLIANT ARTICULATED ROBOT ARM) perteneciente a la Universidad Politécnica Salesiana Campus Sur.

1.2.2. Objetivos específicos.

- Realizar una simulación de los algoritmos de la Cinemática y Dinámica de un manipulador Robótico SCARA, en Matlab.

- Analizar el funcionamiento de la tarjeta SPARTAN 3E.
- Diseñar e Implementar la programación del FPGA basado en VHDL con el desarrollo de algoritmos para el movimiento del brazo robótico.
- Realizar las pruebas necesarias para lograr satisfacer los requerimientos en cuanto a seguridad y confiabilidad del manipulador robótico SCARA.

1.3. Justificación

En la actualidad existe gran disponibilidad de tarjetas de desarrollo como las FPGA´s, y gracias al gran avance tecnológico se propone desarrollar una aplicación en la cual se va a implementar algoritmos cinemáticos para el manejo y control de un brazo robótico tipo Scara.

Estos algoritmos serán implementados en la Tarjeta FPGA Spartan 3E. La tarjeta de desarrollo basada en el FPGA Spartan 3E presenta ventajas debido al gran número de pines de entrada/salida, a la velocidad de procesamiento, capacidad de implementar algoritmos sobre hardware y velocidad de transmisión de datos.

Se desea implementar este proyecto de investigación con el fin de ofrecer una alternativa de control al brazo robótico Scara existente en el laboratorio de la UPS Campus Sur. Con este proyecto se brindará al robot una nueva interfaz de comunicación y control, dotando al laboratorio de más herramientas a nivel de hardware, constituyéndose como complemento didáctico en el estudio de los brazos robóticos.

Esta nueva interfaz propuesta tendrá su manual de operación y uso, permitiendo a los estudiantes de las materias afines realizar prácticas o realizar futuras investigaciones.

1.4. Alcance

En este proyecto de investigación se implementará algoritmos de cinemática directa e inversa para el manipulador robótico Scara perteneciente a la UPS campus SUR, con el uso de la tarjeta FPGA Spartan 3E que será programada mediante el software XILINX y el lenguaje de programación VHDL.

Se utilizará un PC para la visualización de la interfaz gráfica entre la tarjeta FPGA y el manipulador robótico con un protocolo de comunicación serial entre estas.

CAPÍTULO 2
ESTADO DEL ARTE

En este capítulo se analiza el estado del arte del proyecto con respecto al manipulador robótico que en la actualidad se encuentra en inactividad, además se presentará el marco lógico respectivo en el que se basa el diseño e implementación del algoritmo de cinemática directa e inversa utilizando la tarjeta Spartan 3E de Xilinx.

2.1. Situación actual

El proyecto se encuentra en las instalaciones de la Universidad Politécnica Salesiana del Campus Sur, teniendo en cuenta que posee un sistema eléctrico y electrónico a base de microcontroladores con los que se alimenta al manipulador y se realiza el control.

Es un sistema que presenta dificultades al momento de realizar los movimientos con insuficiencia de exactitud y velocidad de transmisión de datos entre la interfaz de control y el manipulador robótico.

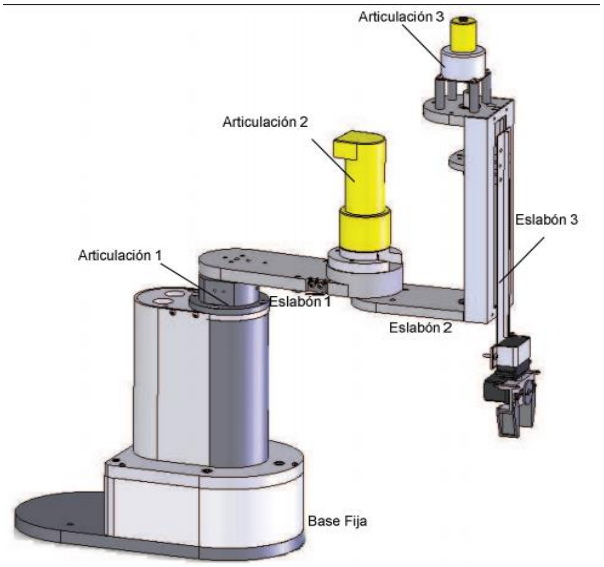
La conformación de los sistemas eléctricos y electrónicos se llevó a cabo siguiendo normas y criterios de diseño, de acuerdo a los requerimientos del robot Scara.

2.2. Referencias del proyecto

Robot de tres grados de libertad tipo Scara (selective compliance assembly robot arm)

Es un brazo robótico que cuenta con tres eslabones unidos por dos articulaciones de tipo revoluta y una prismática, las cuales le proporcionan movilidad en los tres planos. Apoyándose en la modelación cinemática directa e inversa de un robot de tres grados de libertad junto con técnicas de control PID permiten obtener en todo momento posiciones y trayectorias deseadas que serán definidas por el usuario desde una interfaz gráfica o HMI desarrollada en programación de alto nivel, como lo es la plataforma computacional LabVIEW. (Boada & Morales, 2010, pág. 12)

Figura 1. Robot de tres grados de libertad tipo Scara



Fuente: (Boada & Morales, 2010, pág. 21)

Rhino SCARA

Es un brazo robótico de cuatro grados de libertad con una pinza de accionamiento eléctrico. Puede ser operado ya sea desde el controlador Mark III o Mark IV. El Rhino SCARA es un SCARA relativamente grande, es adecuado para operaciones de montaje diseñado en torno a los componentes de Rhino. (rhinorobotics LTD, 2013)

El Mark III es un controlador de 8 ejes con la capacidad de interactuar con 8 líneas de entrada de nivel TTL y puede controlar 8 líneas de salida de nivel TTL. También tiene la capacidad de controlar dos puertos auxiliares que proporcionan 20 voltios cada uno a 1,5 amperios. El controlador Mark III integra en una sola unidad fuentes de alimentación, dispositivos de comunicación, lógica del microprocesador, capacidad de entrada/salida y un lenguaje de software. Se utiliza con la serie de Rhino XR-3 o un robot SCARA. (rhinorobotics LTD, 2013)

Figura 2. Robot Rhino Scara

Imagen: Alejandra Ortiz & Edison Ponce

A continuación se detalla las dimensiones que posee el robot Rhino Scara con sus ángulos y distancias de desplazamiento.

Figura 3. Área de trabajo del robot Rhino Scara

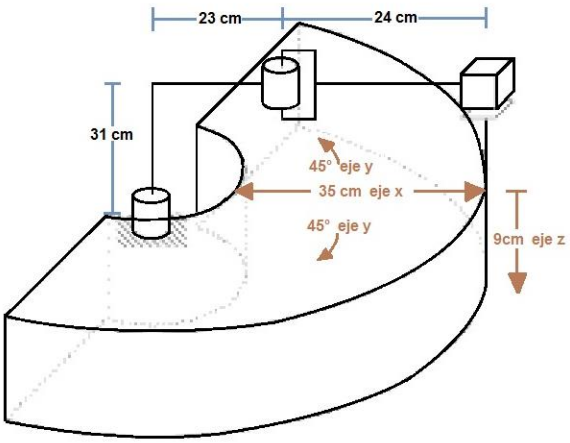


Imagen: Alejandra Ortiz & Edison Ponce

2.3. Los FPGA's

Un FPGA (Field Programmable Gate Array) es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad se puede programar. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip. (CEM-bot, 2013)

Los FPGA’s tienen la ventaja de ser reprogramables (lo que aumenta una enorme flexibilidad al flujo de diseño), los circuitos se “ejecutan” más rápido que en otros dispositivos ya que su ejecución es en paralelo, por lo que los circuitos no necesitan competir por los mismos recursos. Cada tarea de procesos se asigna a una sección dedicada del dispositivo y puede ejecutarse de manera autónoma sin ser afectada por otros bloques de lógica. Como resultado, el rendimiento de una parte de la aplicación no se ve afectado cuando se agregan otros procesos.

Los FPGA’s son el resultado de la convergencia de dos tecnologías diferentes, los dispositivos lógicos programables (PLDs [Programmable Logic Devices]) y los circuitos integrados de aplicación específica (ASIC [application-specific integrated circuit]).
La tarea para programar una FPGA primero es definir la función lógica que realizará. Para ello se tiene entornos de desarrollo especializados en el diseño de sistemas a implementarse en un FPGA. Un diseño puede ser capturado ya sea como esquemático, o haciendo uso de un lenguaje de programación especial. Estos lenguajes de programación especial son conocidos como HDL o Hardware Description Language (Lenguajes de Descripción de Hardware). Los HDLs más utilizados son:

1. VHDL
2. Verilog
3. ABEL

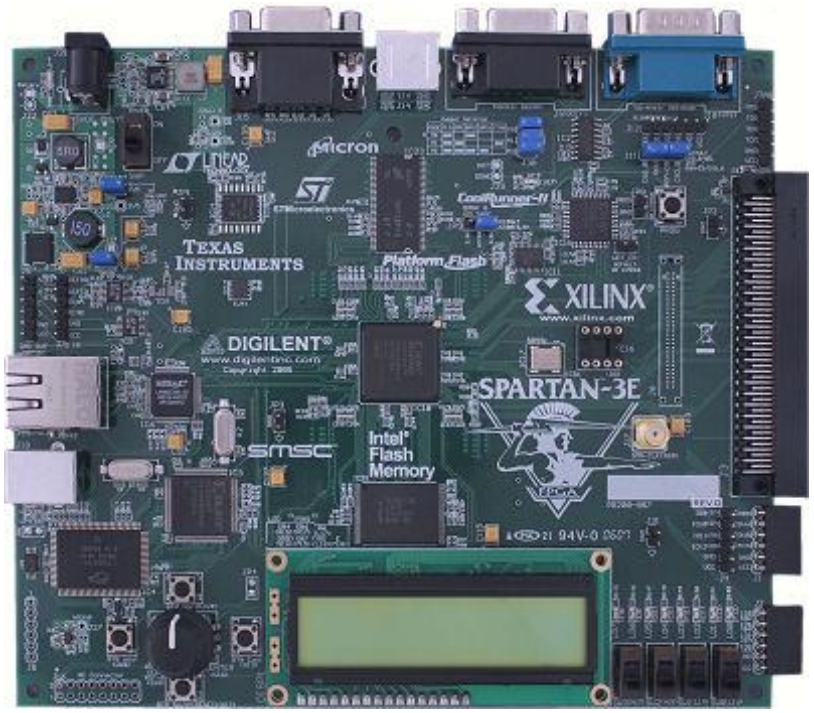
Un FPGA es un circuito integrado que contiene componentes lógicos e interconexiones, ambas programables. Los componentes lógicos pueden ser programados para duplicar la funcionalidad de compuertas lógicas básicas o se pueden crear funciones combinacionales más complejas tales como decodificadores o funciones matemáticas simples.

También se pueden incluir elementos de memoria, los cuales pueden ser simples flip-flops o bloques de memoria más elaborados. La tendencia de los FPGAs ha sido ahora integrar procesadores como el Picoblaze, que puede ser expandido para tener algunos recursos que tienen los microcontroladores. (Marín, 2013)

2.3.1. Descripción de la tarjeta FPGA Spartan 3E Starter.

Se emplea como elemento base un kit de desarrollo FPGA comercial con referencia HWSPAR3E- SK-US de la familia Spartan 3E de Xilinx.

Figura 4. Tarjeta FPGA Spartan 3E Starter



Fuente: (Inc, Xilinx, 2011, pág. 1)

A continuación de describe varias características de esta tarjeta.

- 1 FPGA XC3S500E de la familia Spartan 3E, con más de 232 pines de uso como E/S. Más de 10000 celdas lógicas.
- 1 Memoria PROM con plataforma flash configurable 4 Mbit.
- 1 DDR SDRAM de 64 Mbyte a 100 MHz.
- 1 Memoria flash de 16 Mbyte (128 Mbits) para almacenar la configuración del FPGA o el código del microprocesador.
- 1 SPI serial flash de 16 Mbits (STMicro) que permite almacenar la configuración del FPGA o el código del microprocesador.
- 1 Display LCD de 12-lineas y 16 caracteres.
- 1 Puerto PS/2 para mouse o teclado.
- 1 Puerto para monitor VGA.

- 1 Puerto 10/100 Ethernet.
- 2 Puertos de 9 pines RS-232 (estilo DTE y DCE).
- 1 Puerto USB para FPGA/CPLD con descarga/depuración respectivamente.
- 1 Oscilador de 50 MHz.
- 1 Memoria EEPROM para almacenar archivos de configuración.
- 1 Conector de expansión (FX2 Hirose).
- 3 Conectores de expansión de 6 pines (Diligent).
- 4 Salidas para conversor digital-análogo DAC base SPI con entradas para conversor análogo a digital ADC base SPI con preamplificador de ganancia programable.
- 1 Encoder rotacional con botón pulsador central.
- 8 Salidas digitales visualizadas en LED´s.
- 4 Swiches deslizables.
- 4 Swiches pulsadores.
- 1 Entrada para reloj.
- 1 Socket DIP 8 pines para una señal de reloj auxiliar.

2.3.2. Funcionamiento de la tarjeta FPGA Spartan 3E Starter.

Los FPGA Spartan 3E de Xilinx están conformadas por un conjunto de Bloques Lógicos Configurables (Configurable Logic Blocks: CLBs) rodeados por un perímetro de Bloques Programables de entrada/salida (Programmable Input/Output Blocks: IOBs). Estos elementos funcionales están interconectados por una jerarquía de canales de conexión (Routing Channels), la que incluye una red de baja capacitancia para la distribución de señales de reloj de alta frecuencia. Adicionalmente el dispositivo cuenta con 24 bloques de memoria RAM de 2Kbytes de doble puerto, cuyos anchos de buses son configurables, y con 12 bloques de multiplicadores dedicados de 18 X 18 bits. (U-Cursos, 2006)

Los cinco elementos funcionales programables que lo componen son los siguientes:

i. Bloques de entrada/salida (Input/Output Blocks – IOBs): Controlan el flujo de datos entre los pines de entrada/salida y la lógica interna del dispositivo. Soportan flujo bidireccional más operación tri-estado y un conjunto de

estándares de voltaje e impedancia controlados de manera digital. (U-Cursos, 2006)

ii. Bloques Lógicos Configurables (Configurable Logic Blocks – CLBs): contienen Look-Up Tables (Tablas de consulta: son la forma en que su lógica en realidad se implementa) o (LUTs) basadas en tecnología RAM para implementar funciones lógicas y elementos de almacenamiento que pueden ser usados como flip-flops o como latches (circuito electrónico usado para almacenar información en sistemas lógicos asíncronos). (U-Cursos, 2006)

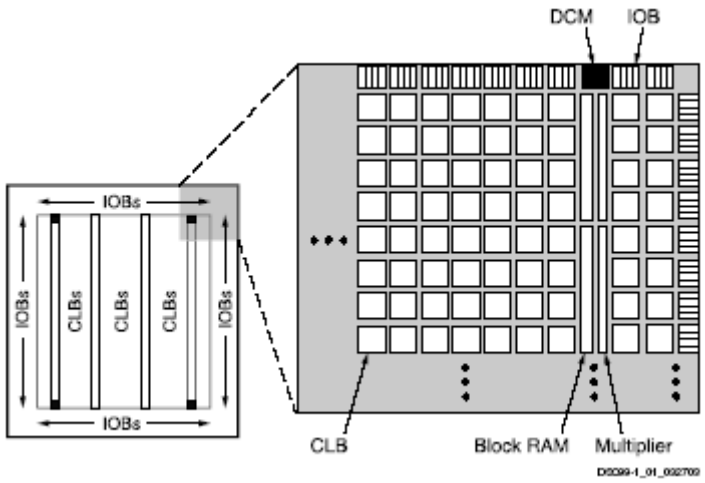
iii. Bloques de Memoria RAM (Block RAM): proveen almacenamiento de datos en bloques de 18 Kbits con dos puertos independientes cada uno. (U-Cursos, 2006)

iv. Bloques de multiplicación que aceptan dos números binarios de 18 bit como entrada y entregan uno de 36 bits. (U-Cursos, 2006)

v. Administradores Digitales de Reloj (Digital Clock Managers – DCMs): estos elementos proveen funciones digitales auto-calibradas, las que se encargan de distribuir, retrasar arbitrariamente en pocos grados, desfasar en 90, 180 y 270 grados, dividir y multiplicar las señales de reloj de todo el circuito. (U-Cursos, 2006)

Los elementos descritos están organizados como se muestra en la figura 5. Un anillo de IOBs rodea un arreglo regular de CLBs. Atraviesa este arreglo una columna de Bloques de memoria RAM, compuesta por varios bloques de 18 Kbit, cada uno de los cuales está asociado con un multiplicador dedicado. Los DCMs están colocados en los extremos de dichas columnas.

Figura 5. Arquitectura de la Spartan 3E



Fuente: (Inc, Xilinx, 2013, pág. 3)

A continuación se hace una descripción más detallada de cada uno de los elementos funcionales del FPGA, y luego se describe el proceso de configuración de la misma.

2.3.2.1. Bloques de entrada/salida IOB.

Los bloques de entrada/salida (IOB) suministran una interfaz bidireccional programable entre un pin de entrada/salida y la lógica interna del FPGA. Un diagrama simplificado de la estructura interna de un IOB aparece en la figura 6. (U-Cursos, 2006)

Hay tres rutas para señales en un IOB: la ruta de salida, la ruta de entrada y la ruta tri-estado. Cada ruta tiene su propio par de elementos de almacenamiento que pueden actuar tanto como registros o como latches (circuito electrónico usado para almacenar información en sistemas lógicos asíncronos). Las tres rutas principales son:

- La ruta de entrada, que lleva datos desde el pad, que está unido al pin del package, a través de un elemento de retardo opcional programable, directamente a la línea I. Después del elemento de retardo hay rutas alternativas a través de un par de elementos de almacenamiento hacia las líneas IQ1 e IQ2. Las tres salidas del IOB todas conducen a la lógica interna del FPGA. (U-Cursos, 2006)

- La ruta de salida, que parte con las líneas O1 y O2, lleva datos desde la lógica interna del FPGA, a través de un multiplexor y del driver tri-estado hacia el pad del IOB. En suma a esta ruta directa, el multiplexor da la opción de insertar un par de elementos de almacenamiento. (U-Cursos, 2006)
- La ruta tri-estado determina cuando el driver de salida está en alta impedancia. (U-Cursos, 2006)

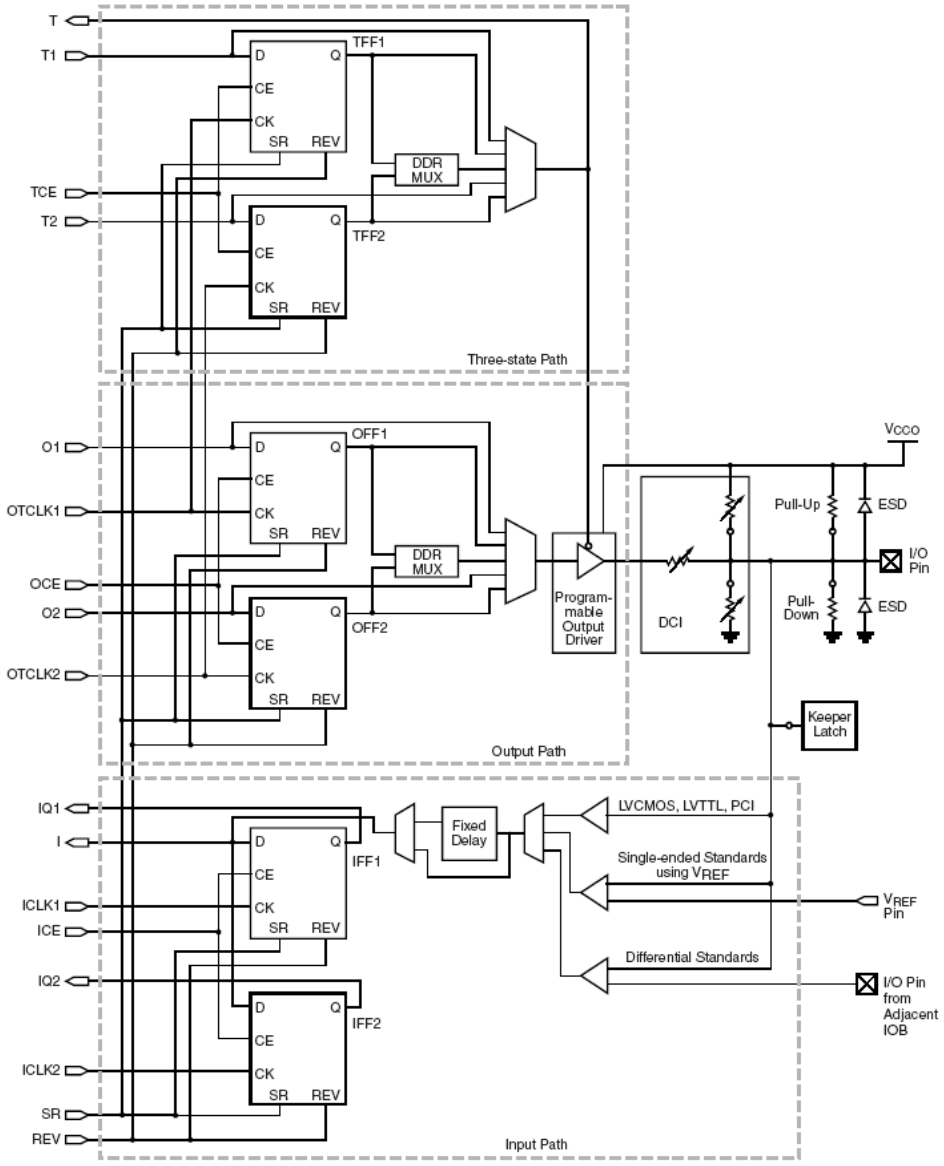
Las líneas T1 y T2 llevan datos desde la lógica interna a través de un multiplexor hacia el driver de salida. En suma a esta ruta directa, el multiplexor da la opción de entregar un par de elementos de almacenamiento. (U-Cursos, 2006)

Todas las rutas de señales que entran al IOB, incluidas aquellas asociadas con los elementos de almacenamiento tienen una opción de inversión. Cualquier inversor colocado (en la programación) en estas rutas es automáticamente absorbido dentro del IOB. (U-Cursos, 2006)

Hay tres pares de elementos de almacenamiento en cada IOB, un par para cada uno de las tres rutas. Es posible configurar cada uno de esos elementos como un flip-flop "D" gatillado por flanco (FD) o como un latch sensible a nivel (LD). Estos elementos son controlados con la misma red de distribución de relojes que se utiliza para todo el sistema. (U-Cursos, 2006)

El par de elementos de almacenamiento tanto de la ruta de salida o del driver tri-estado pueden ser usados en conjunto, con un multiplexor especial para producir transmisión de doble tasa de datos (DDR). Esto se logra tomando datos sincronizados con el flanco de subida del reloj y convirtiéndolos en bits sincronizados tanto con el flanco de subida como con el de bajada. A esta combinación de dos registros y un multiplexor se le llama flip-flop tipo D de doble tasa de datos (FDDR). (U-Cursos, 2006)

Figura 6. Diagrama simplificado de un IOB de la Spartan 3E



Fuente: (Inc, Xilinx, 2013, pág. 11)

Cada IOB cuenta además con otros elementos, entre los cuales cuentan las resistencias de Pull-Up y de Pull-Down, que tienen el objetivo de establecer niveles altos o bajos respectivamente en las salidas de los IOBs que no están en uso; un circuito de retención (Keeper) del último nivel lógico que se mantiene, después de que todos los drivers han sido apagados, lo que es útil para cuidar que las líneas de un bus no floten, cuando los drivers conectados están en alta impedancia se habilita

un circuito de protección para descargas electro estáticas (protección ESD), que utiliza diodos de protección. (U-Cursos, 2006)

Finalmente cada IOB cuenta con un control para el slew rate (incapacidad de un amplificador para seguir variaciones rápidas de la señal de entrada) y para la corriente de salida máxima. El primero otorga la posibilidad de elegir una tasa alta de cambio de nivel (con bajo slew rate) o una tasa máxima menor para la utilización de los puertos en la integración a buses, donde al pasar de alta impedancia a un nivel de voltaje suele producirse transiciones inesperadas. El segundo entrega siete niveles diferentes de corrientes máximas tanto para el estándar CMOS como para el TTL, lo que permite adaptarse a dispositivos que necesitan mayores corrientes para su activación, en el caso del estándar LVCMOS a 2.5V el rango de corrientes es de 2 a 24 mA (2, 4, 6, 8, 12, 16, 24 mA). (U-Cursos, 2006)

Los IOB soportan 17 estándares de señales de salida de terminación única y seis de señal diferencial, también cuentan con un sistema integrado, para coincidir con la impedancia de las líneas de transmisión que llegan al FPGA, llamado Control Digital de Impedancia (DCI), el que permite elegir hasta 5 tipos diferentes de terminaciones, utilizando una red de resistencias internas que se ajustan en serie o en paralelo, dependiendo de las necesidades del estándar elegido. (U-Cursos, 2006)

2.3.3. Spartan-6 XC6SLX150T.

A continuación se detalla algunas características de esta tarjeta.

Conectores I / O

- Dos conectores FMC LPC para fines generales de expansión de E/S.
- Un conector para una tarjeta SD.
- Conector Avnet LCD Interface (ALI).

Conectores RocketIO ™ GTP transceptor

- Un conector pequeño de jaula (SFP).
- Dos transceptores suministrados en un conector FMC para el uso de un módulo de ampliación.
- Una PCI Express de complemento de interfaz de tarjeta a 2.5 Gbps.
- Un conector de host SATA.

Memoria

- 128 MB de SDRAM DDR3.
- 32 MB FLASH.

Comunicación

- Puerto serial RS- 232.
- USB 2.0.
- Puerto USB -RS232.
- Puerto Ethernet 10/100/1000 Mbits.

Potencia

- Tensiones de alimentación reguladas en 5.0, 3.3, 2.5, 1.8, 1.5 y 1.2 V, derivados de la ranura PCI Express o una alimentación externa de 12 V.
- Punto de reguladores de carga para carriles de alimentación MGT.

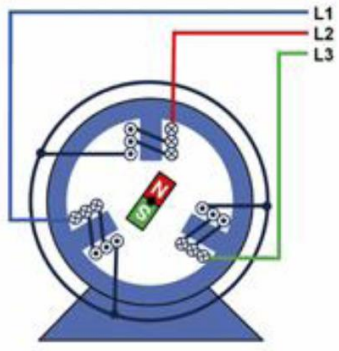
Configuración

- Plataforma Flash de configuración XCF32 y XCF08.
- Cable IV Xilinx Parallel o plataforma con soporte USB para el conector JTAG de Programación/configuración.

2.4. Motor brushless

Como su propio nombre lo indica, brushless quiere decir "sin escobillas". En este tipo de motor la corriente eléctrica pasa directamente por los bobinados del estator o carcasa, por lo tanto aquí no son necesarios ni las escobillas ni el colector que se utilizan en los brushed. Esta corriente eléctrica genera un campo electromagnético que interacciona con el campo magnético creado por los imanes permanentes del rotor, haciendo que aparezca una fuerza que hace girar al rotor y por lo tanto al eje del motor. (Arian, 2012).

Figura 7. Diagrama esquemático de un motor brushless



Fuente: (Heredia, 2014, pág. 41)

En la realización de este proyecto se ha utilizado los motores reductores cilíndricos serie GM8000 de Pittman que cuentan con engranajes de acero sinterizado, están disponibles en tres longitudes de pila y cuentan con terminales de salida lateral.

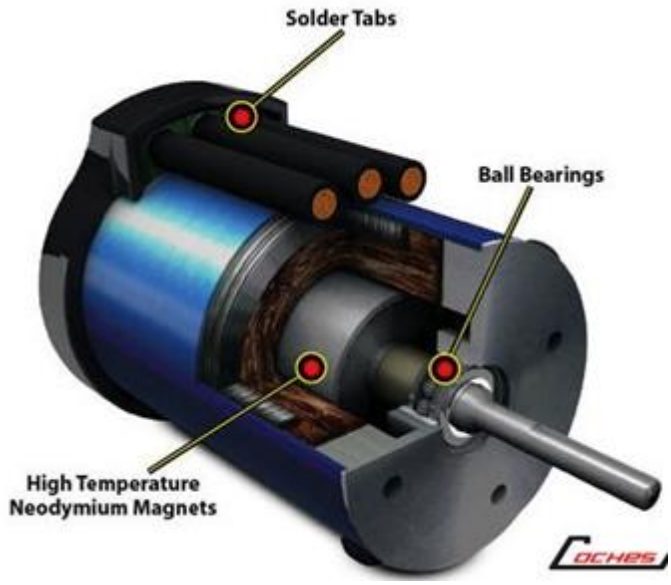
Los motores con caja de reducción Pittman brindan un funcionamiento suave, silencioso y de larga vida. Las armaduras están sesgadas para minimizar el rizo de par, incluso a bajas velocidades y los arrollamientos son de resina impregnada de una mayor fiabilidad en aplicaciones de movimiento incremental. (Heredia, 2014, pág. 41)

Las características por las cuales se eligió estos motores son la calidad de sus componentes, como los rodamientos de bolas de alta fiabilidad y engranajes de alto rendimiento.

2.4.1. Componentes del motor brushless.

Los motores brushless están compuestos por una parte móvil que es el rotor, donde se encuentran los imanes permanentes, y una parte fija, denominada estator o carcasa, sobre la cual van dispuestos los bobinados de hilo conductor. (Arian, 2012)

Figura 8. Componentes del motor brushless



Fuente: (Arian, 2012)

2.5. Cinemática del robot

La cinemática del robot es el estudio de su movimiento con respecto a un sistema de referencia.

Los diferentes sistemas de referencia pueden ser descritos mediante:

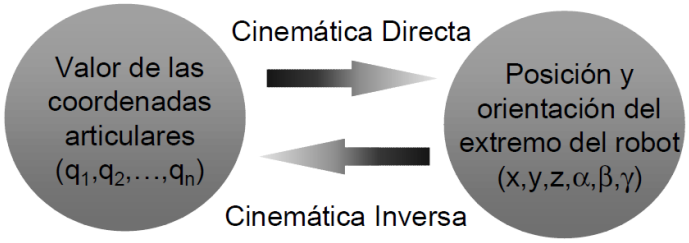
- Modelos analíticos del movimiento espacial en función del tiempo.
- Relaciones de localización del extremo del robot-valores articulares. (Hernández, 2013)

Las cinemáticas que pueden surgir en un robot pueden ser de dos tipos con un modelo diferencial entre ambas.

- Cinemática directa: Se ocupa en determinar la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas de referencia, conocidos los ángulos de las articulaciones y los parámetros geométricos de los elementos del robot. (Hernández, 2013)
- Cinemática inversa: Se ocupa en determinar la configuración que debe adoptar el robot para una posición y orientación del extremo conocidas. (Hernández, 2013)

- Modelo diferencial (Matriz Jacobiana): Es la matriz formada por las relaciones entre las velocidades de movimiento de las articulaciones y las del extremo del robot. (Hernández, 2013)

Figura 9. Relación entre cinemática directa e inversa



Fuente: (Hernández, 2013, pág. 4)

2.5.1. Cinemática directa del robot Scara.

La cinemática directa se encarga de determinar la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas de referencia, conocidos los ángulos de las articulaciones y los parámetros geométricos de los elementos del robot, se logra a continuación detallar sus funciones. (Hernández, 2013)

$$x = f_x(q_1, q_2, q_3, q_4, q_5, q_6)$$
$$y = f_y(q_1, q_2, q_3, q_4, q_5, q_6)$$
$$z = f_z(q_1, q_2, q_3, q_4, q_5, q_6)$$
$$\alpha = f_\alpha(q_1, q_2, q_3, q_4, q_5, q_6)$$
$$\beta = f_\beta(q_1, q_2, q_3, q_4, q_5, q_6)$$
$$\gamma = f_\gamma(q_1, q_2, q_3, q_4, q_5, q_6)$$

Dónde:

q_{n-1} = Son las variables de las articulaciones.

Para articulaciones revolutas (unión que gira sobre un único eje con respecto al otro) las variables son ángulos.

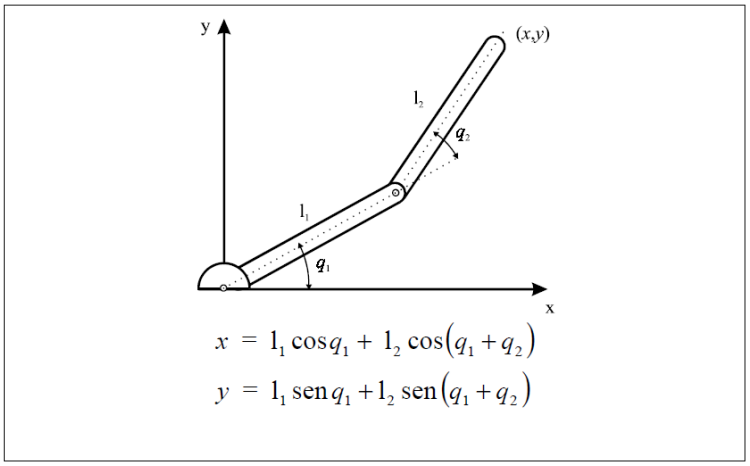
Para articulaciones prismáticas (uniones arraigadas que se desplazan dentro y a lo largo de cada una de éstas) las variables son distancias.

x, y, z = Coordenadas de la posición del extremo del robot.

α, β, ϒ = Ángulos de la orientación del extremo del robot.

La obtención de estas relaciones no es en general complicada, siendo incluso en ciertos casos (robots de pocos grados de libertad) fácil de encontrar mediante simples consideraciones geométricas. (Hernández, 2013)

Figura 10. Por ejemplo para el caso de un robot con 2 grados de libertad



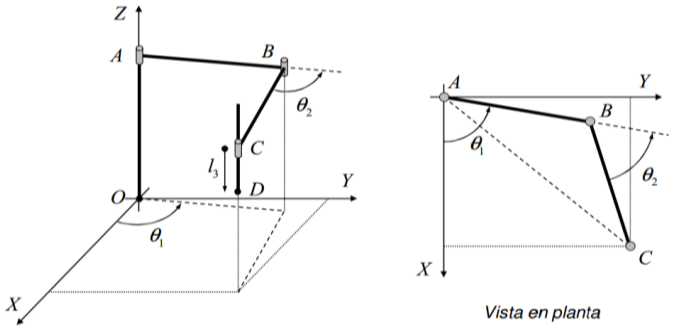
Fuente: (Hernández, 2013, pág. 6)

En general, un robot de n grados de libertad está formado por n eslabones unidos por n articulaciones, de forma que cada par articulación-eslabón constituye un grado de libertad. A cada eslabón se le puede asociar un sistema de referencia a él y utilizando las transformaciones homogéneas, es posible representar las rotaciones y traslaciones relativas entre los distintos eslabones que componen el robot. (Hernández, 2013)

2.5.2. Cinemática inversa del robot Scara.

Para formular la cinemática inversa del robot Scara se define el sistema de referencia que se muestra a continuación. (Noriega, Muñiz, & García, 2010)

Figura 11. Esquema del robot Scara



Fuente: (Noriega, Muñiz, & García, 2010, pág. 5)

Para realizar la modelización es necesario conocer las longitudes l_{OA}, l_{AB} y l_{BC} así como el modo de ensamblaje deseado para el robot. Como se va a resolver la cinemática inversa, se dispone de la posición (X_D, Y_D, Z_D), la velocidad $(\dot{X}_D, \dot{Y}_D, \dot{Z}_D)$ y la aceleración $(\ddot{X}_D, \ddot{Y}_D, \ddot{Z}_D)$ del punto D del robot Scara y se desea calcular, en cada instante temporal, el ángulo θ_1 y sus derivadas temporales ($\dot{\theta}_1$ y $\ddot{\theta}_1$), el ángulo θ_2 y sus derivadas temporales ($\dot{\theta}_2 y\ \ddot{\theta}_2$) y la distancia l_3 y sus derivadas temporales ($\dot{l_3}$ y $\ddot{l_3}$). (Noriega, Muñiz, & García, 2010)

Las coordenadas del punto O y sus derivadas temporales son nulas por estar situado en el origen y ser un punto fijo tal y como se muestra en la ecuación 1. (Noriega, Muñiz, & García, 2010)

Ecuación 1. Coordenadas del punto O

$$x_O = y_O = z_O = 0$$
$$\dot{x}_O = \dot{y}_O = \dot{z}_O = 0$$
$$\ddot{x}_O = \ddot{y}_O = \ddot{z}_O = 0$$

Las coordenadas del punto A también son constantes y su expresión se muestra en la ecuación 2. (Noriega, Muñiz, & García, 2010)

Ecuación 2. Coordenadas del punto A

$$x_A = x_O \qquad y_A = y_O \qquad z_A = z_O + l_{OA}$$
$$\dot{x}_A = \dot{y}_A = \dot{z}_A = 0$$
$$\ddot{x}_A = \ddot{y}_A = \ddot{z}_A = 0$$

Las coordenadas del punto C se obtienen a partir de los puntos A y D y su expresión se muestra en la ecuación 3. (Noriega, Muñiz, & García, 2010)

Ecuación 3. Coordenadas del punto C

$$x_C = x_D \qquad y_C = y_D \qquad z_C = z_A$$
$$\dot{x}_C = \dot{x}_D \qquad \dot{y}_C = \dot{y}_D \qquad \dot{z}_C = 0$$
$$\ddot{x}_C = \ddot{x}_D \qquad \ddot{y}_C = \ddot{y}_D \qquad \ddot{z}_C = 0$$

Entonces, los valores de $l_3, \dot{l_3}, \ddot{l_3}$ se pueden calcular directamente como se muestra en la ecuación 4. (Noriega, Muñiz, & García, 2010)

Ecuación 4. Obtención de los valores $l_3, \dot{l_3}, \ddot{l_3}$

$$l_3 = z_A - z_D$$
$$\dot{l}_3 = -\dot{z}_D$$
$$\ddot{l}_3 = -\ddot{z}_D$$

Para obtener la posición del punto B, se tiene en cuenta que A, B y C están contenidos en un mismo plano paralelo al XOY donde $Z_B = Z_A$. Además, dicho plano no varía su coordenada Z por lo que $\dot{Z}_B = \ddot{Z}_B = 0$. (Noriega, Muñiz, & García, 2010)

Una vez conocidas la posición, velocidad y aceleración en X e Y de los puntos A y B, se puede calcular el ángulo θ_{BC}, y sus dos primeras derivadas ($\dot{\theta}_{BC}$ y $\ddot{\theta}_{BC}$) respecto del tiempo. Finalmente se calcula su ángulo Θ_2 y sus derivadas temporales se pueden calcular mediante la siguiente ecuación (Noriega, Muñiz, & García, 2010)

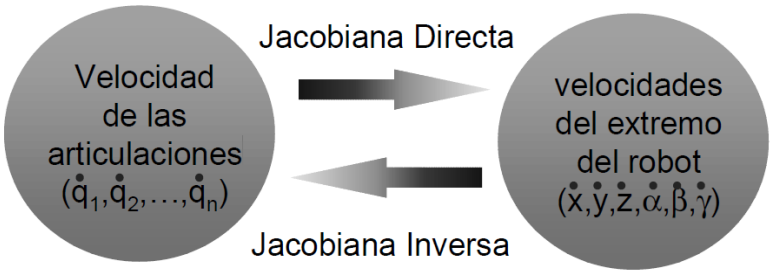
Ecuación 5. Obtención de los valores θ_2, $\dot{\theta}_2$ y $\ddot{\theta}_2$

$$\theta_2 = \theta_1 - \theta_{BC}$$
$$\dot{\theta}_2 = \dot{\theta}_1 - \dot{\theta}_{BC}$$
$$\ddot{\theta}_2 = \ddot{\theta}_1 - \ddot{\theta}_{BC}$$

2.6. Matriz Jacobiana

Es el medio que permite conocer las velocidades del extremo del robot a partir de las velocidades de cada articulación. (Hernández, 2013)

Figura 12. Matriz Jacobiana



Fuente: (Hernández, 2013, pág. 28)

Con las funciones de x,y,z correspondientes a las posiciones y su primera derivada correspondiente a las velocidades se puede obtener la matriz Jacobiana al realizar el producto punto entre la matriz J (derivada de las posiciones con respecto a cada punto) y la matriz de las velocidades de las articulaciones, como se detalla en las siguientes figuras.

Figura 13. Relaciones diferenciales

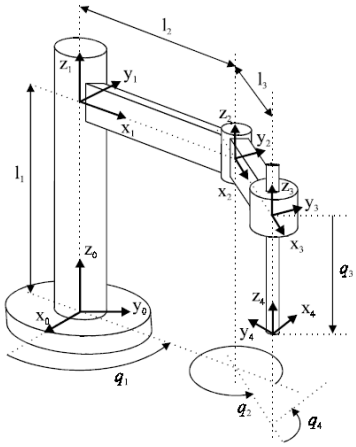
$$x = f_x(q_1,\ldots,q_n) \qquad y = f_y(q_1,\ldots,q_n) \qquad z = f_z(q_1,\ldots,q_n)$$
$$\alpha = f_\alpha(q_1,\ldots,q_n) \qquad \beta = f_\beta(q_1,\ldots,q_n) \qquad \gamma = f_\gamma(q_1,\ldots,q_n)$$

$$\dot{x} = \sum_1^n \frac{\partial f_x}{\partial q_i}\dot{q}_i \qquad \dot{y} = \sum_1^n \frac{\partial f_y}{\partial q_i}\dot{q}_i \qquad \dot{z} = \sum_1^n \frac{\partial f_z}{\partial q_i}\dot{q}_i$$
$$\dot{\alpha} = \sum_1^n \frac{\partial f_\alpha}{\partial q_i}\dot{q}_i \qquad \dot{\beta} = \sum_1^n \frac{\partial f_\beta}{\partial q_i}\dot{q}_i \qquad \dot{\gamma} = \sum_1^n \frac{\partial f_\gamma}{\partial q_i}\dot{q}_i$$

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ \dot{\alpha}\\ \dot{\beta}\\ \dot{\gamma}\end{bmatrix} = \mathbf{J}\cdot\begin{bmatrix}\dot{q}_1\\ \vdots\\ \vdots\\ \dot{q}_n\end{bmatrix} \qquad \text{con} \quad \mathbf{J} = \begin{bmatrix}\frac{\partial f_x}{\partial q_1} & \cdots & \frac{\partial f_x}{\partial q_n}\\ \vdots & \ddots & \vdots\\ \frac{\partial f_\gamma}{\partial q_1} & \cdots & \frac{\partial f_\gamma}{\partial q_n}\end{bmatrix}$$

Fuente: (Hernández, 2013, pág. 29)

Figura 14. Matriz Jacobiana de un robot Scara



$$x = l_3C_{12} + l_2C_1$$
$$y = l_3S_{12} + l_2S_1$$
$$z = l_1 - q_3$$

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = \begin{bmatrix}-(l_3S_{12} + l_2S_1) & -l_3S_{12} & 0\\ l_3C_{12} + l_2C_1 & l_3C_{12} & 0\\ 0 & 0 & -1\end{bmatrix}\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\end{bmatrix}$$

Fuente: (Hernández, 2013, pág. 30)

2.6.1. Métodos de cálculo de la matriz Jacobiana inversa.

Existen varios métodos entre estos están:

- Inversión simbólica de la matriz Jacobiana
  - ❖ Gran complejidad (matriz 6x6)
- Evaluación numérica de J e inversión numérica
  - ❖ Necesidad de recómputo continuo
  - ❖ En ocasiones J no es matriz cuadrada
  - ❖ En ocasiones el determinante de la matriz es | J | = 0
- A partir del modelo cinemático inverso, como se observa en la siguiente figura. (Hernández, 2013)

Figura 15. Modelo cinemático de la matriz Jacobiana inversa

$$q_1 = \mathrm{f}_1(x, y, z, \alpha, \beta, \gamma)$$
$$\vdots$$
$$q_n = \mathrm{f}_n(x, y, z, \alpha, \beta, \gamma)$$

$$\begin{bmatrix} \dot{q}_1 \\ \vdots \\ \vdots \\ \vdots \\ \dot{q}_n \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \dot{x} \\ \vdots \\ \vdots \\ \vdots \\ \dot{\gamma} \end{bmatrix} \qquad \mathbf{J}^{-1} = \begin{bmatrix} \frac{\partial \mathrm{f}_1}{\partial x} & \cdots & \cdots & \cdots & \frac{\partial \mathrm{f}_1}{\partial \gamma} \\ \vdots & \ddots & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ \frac{\partial \mathrm{f}_n}{\partial x} & \cdots & \cdots & \cdots & \frac{\partial \mathrm{f}_n}{\partial \gamma} \end{bmatrix}$$

Fuente: (Hernández, 2013, pág. 31)

2.7. Matrices de transformación A y T

La matriz $^{\mathbf{i-1}}\mathbf{A_i}$ o matriz de transformación homogénea representa la posición y orientación relativa entre los sistemas asociados a dos eslabones consecutivos del robot. Donde i es el número máximo de eslabones del robot.

Ecuación 6. Conexión de matrices A

$$^{0}\mathrm{A}_3 = {}^{0}\mathrm{A}_1\,{}^{1}\mathrm{A}_2\,{}^{2}\mathrm{A}_3$$

La matriz **T** o matriz $^{\mathbf{0}}\mathbf{A_n}$ se aplica al momento de considerar todos los grados de libertad del robot.

Ecuación 7. Matriz T o matriz $^{0}\mathrm{A}_n$

$$\mathrm{T} = {}^{0}\mathrm{A}_6 = {}^{0}\mathrm{A}_1\,{}^{1}\mathrm{A}_2\,{}^{2}\mathrm{A}_3\,{}^{3}\mathrm{A}_4\,{}^{4}\mathrm{A}_5\,{}^{5}\mathrm{A}_6$$

2.8. Algoritmo de Denavit-Hartenberg

Este algoritmo fue creado para definir el paso de un sistema de referencia asociado a una articulación con la siguiente articulación. Sólo dependen de las características geométricas de cada eslabón y de las articulaciones que le unen con el anterior y siguiente eslabón/articulación (no dependen de la posición del robot) y definen las matrices A que permiten el paso de un sistema de referencia asociado a una articulación al siguiente y por tanto definen las matrices T. (Hernández, 2013)

El Algoritmo de Denavit-Hartenberg se basa en:

- Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con n (último eslabón móvil). Se numerará como eslabón 0 a la base fija del robot.
- Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad) y acabando en n.
- Localizar el eje de cada articulación. Si ésta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.
- Para i de 0 a n-1 situar el eje Z_i sobre el eje de la articulación i+1.
- Situar el origen del sistema de la base $\{S_0\}$ en cualquier punto del eje Z_0. Los ejes X_0 e Y_0 se situarán de modo que formen un sistema dextrógiro con Z_0.
- Para i de 1 a n-1, situar el sistema $\{S_i\}$ (solidario al eslabón i) en la intersección del eje Z_i con la línea normal común a Z_{i-1} y Z_i. Si ambos ejes se cortasen se situaría $\{S_i\}$ en el punto de corte. Si fuesen paralelos $\{S_i\}$ se situaría en la articulación i+1.
- Situar X_i en la línea normal común a Z_{i-1} y Z_i
- Situar Y_i de modo que forme un sistema dextrógiro con X_i y Z_i.
- Situar el sistema $\{S_n\}$ en el extremo del robot de modo que Z_n coincida con la dirección de Z_{n-1} y X_n sea normal a Z_{n-1} y Z_n.
- Obtener θ_i como el ángulo que hay que girar en torno a Z_{i-1} para que X_{i-1} y X_i queden paralelos.
- Obtener d_i como la distancia, medida a lo largo de Z_{i-1}, que habría que desplazar $\{S_{i-1}\}$ para que X_i y X_{i-1} quedasen alineados.

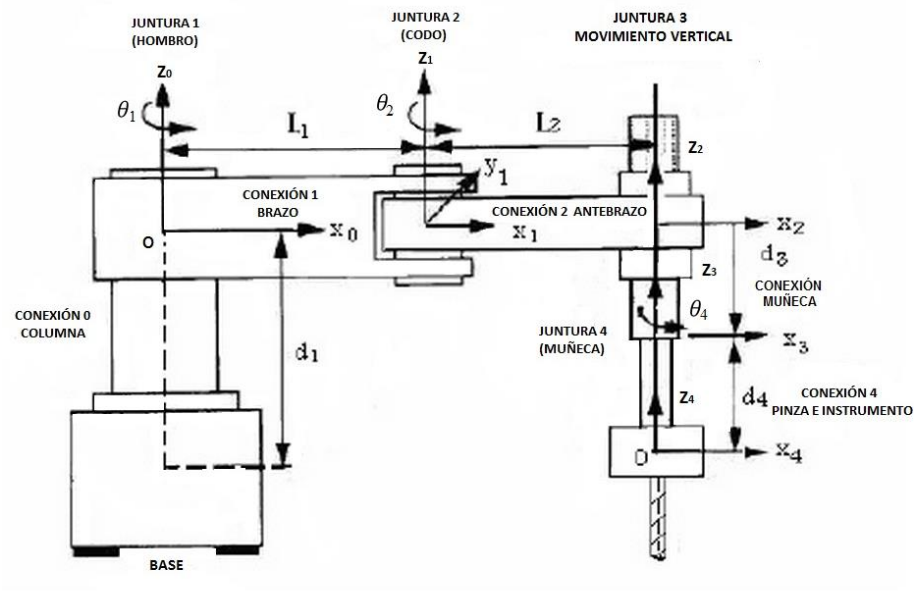
- Obtener a_i como la distancia medida a lo largo de X_i (que ahora coincidiría con X_{i-1}) que habría que desplazar el nuevo $\{S_{i-1}\}$ para que su origen coincidiese con $\{S_i\}$.
- Obtener α_i como el ángulo que habría que girar entorno a X_i (que ahora coincidiría con X_{i-1}), para que el nuevo $\{S_{i-1}\}$ coincidiese totalmente con $\{S_i\}$.
- Obtener las matrices de transformación ${}^{i-1}A_i$
- Obtener la matriz de transformación entre la base y el extremo del robot $T = {}^{0}A_1 \, {}^{1}A_2 \, ... \, {}^{n-1}A_n$.
- La matriz T define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido a la base en función de las n coordenadas articulares. (Hernández, 2013)

2.9. Desarrollo matemático de la cinemática directa e inversa en Matlab

Los parámetros del algoritmo Denavit-Hartenberg (D-H) utilizados en el robot Scara especificados en la siguiente figura se encuentran definidos en la tabla 1.

Se detalla únicamente cuadro grados de libertad, debido a que el quinto grado de libertad del robot Scara corresponde a la pinza, con movimientos únicos en el eje x, como se muestra en la siguiente figura.

Figura 16. Parámetros del robot Scara



Fuente: (Research, 2012)
Elaborado por: Alejandra Ortiz & Edison Ponce

Tabla 1. *Parámetros denavit-hartenberg (D-H) del robot*

i	$\boldsymbol{\theta_i}$	$\boldsymbol{d_i}$	$\boldsymbol{a_i}$	$\boldsymbol{\alpha_i}$
1	θ_1	0	L_1	0
2	θ_2	0	L_2	0
3	0	d_3	0	0
4	θ_3	d_4	0	0

Ecuación 8. Resultado de matriz T_1 usando la convención D-H

$$T_1^0 = A_1 = \begin{pmatrix} c_1 & -s_1 & 0 & L_1c_1 \\ s_1 & c_1 & 0 & L_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ecuación 9 . Resultado de matriz T_2 usando la convención D-H

$$T_2^1 = A_2 = \begin{pmatrix} c_2 & -s_2 & 0 & L_2c_2 \\ s_2 & c_2 & 0 & L_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ecuación 10. Resultado de matriz T_3 usando la convención D-H

$$T_3^2 = A_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ecuación 11. Resultado de matriz T_4 usando la convención D-H

$$T_4^3 = A_4 = \begin{pmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & -d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ecuación 12 . Matriz total de transformación

$$T_4^0 = \begin{pmatrix} c_{124} & -s_{124} & 0 & L_2c_{12} + L_1c_1 \\ s_{124} & c_{124} & 0 & L_2s_{12} + L_1S_1 \\ 0 & 0 & 1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Solución inversa para la posición.

Ecuación 13. Ubicación deseada del robot Scara

$$T_H^R = \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ecuación 14. Ecuación final para representación del robot

$$T_H^R = A_1A_2A_3A_4 = T_4^0$$

Para resolver el ángulo θ_4 , ambos lados de la ecuación 14 se multiplica sucesivamente con las matrices, $A_3^{-1}A_2^{-1}A_1^{-1}$ de tal manera que:

Ecuación 15. Obtención de A_4

$$A_3^{-1}A_2^{-1}A_1^{-1}T_H^R = A_4$$

Ecuación 16. Resultado sucesivo de la multiplicación de ambas ecuaciones

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} c_2 & s_2 & 0 & -L_2 \\ -s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} c_1 & s_1 & 0 & -L_1 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} n_xc_{12} + n_ys_{12} & o_xc_{12} + o_ys_{12} & a_xc_{12} + a_ys_{12} & p_xc_{12} + p_ys_{12} - L_1c_2 - L_2 \\ -n_xs_{12} + n_yc_{12} & -o_xs_{12} + o_yc_{12} & -a_xs_{12} + a_yc_{12} & -p_xs_{12} + p_yc_{12} - L_1s_2 \\ n_z & o_z & a_z & p_z + d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ecuación 17. Resultado del elemento 1,4 de la ecuaciones 12 y 13

$$P_x = L_1C_1 + L_2C_{12}$$

Ecuación 18. Resultado del elemento 2,4 de la ecuaciones 12 y 13

$$P_y = L_1S_1 + L_2S_{12}$$

Ecuación 19. Obtención de C_2 de la ecuación 17 y 18

$$C_2 = \frac{1}{2L_1L_2}(P_x^2 + P_y^2 - L_1^2 - L_2^2)$$

Ecuación 20. Obtención de S_2 de la ecuación 17 y 18

$$S_2 = \pm\sqrt{1 - C_2^2}$$

Ecuación 21. Obtención de Θ_2 de la ecuación 17 y 18

$$\theta_2 = tan^{-1}\frac{S_2}{C_2}$$

Ecuación 22. Reordenando la ecuación 17 se obtiene

$$P_x = (L_1 + L_2C_2)C_1 - L_2S_2S_1$$

Ecuación 23. Reordenando la ecuación 18 se obtiene

$$P_y = L_2S_2C_1 + (L_1 + L_2C_2)S_1$$

Ecuación 24. Resolución de ecuaciones 22 y 23 por la regla de Kramer para obtener Θ_1

$$\Delta = \begin{pmatrix} L_1 + L_2C_2 & -L_2S_2 \\ L_2S_2 & L_1 + L_2C_2 \end{pmatrix} = (L_1 + L_2C_2)^2 + (L_2S_2)^2$$

$$\Delta S_1 = \begin{pmatrix} L_1 + L_2C_2 & P_x \\ L_2S_2 & P_y \end{pmatrix} = (L_1 + L_2C_2)P_y - (L_2S_2)P_x$$

$$\Delta C_1 = \begin{pmatrix} P_x & -L_2S_2 \\ P_y & L_1 + L_2C_2 \end{pmatrix} = (L_1 + L_2C_2)P_x + L_2S_2P_y$$

$$S_1 = \frac{\Delta S_1}{\Delta} = \frac{(L_1 + L_2C_2)P_y - L_2S_2P_x}{(L_1 + L_2C_2)^2 + (L_2S_2)^2} = \frac{(L_1 + L_2C_2)P_y - L_2S_2P_x}{{P_x}^2 + {P_y}^2}$$

$$C_1 = \frac{\Delta C_1}{\Delta} = \frac{(L_1 + L_2C_2)P_x + L_2S_2P_y}{(L_1 + L_2C_2)^2 + (L_2S_2)^2} = \frac{(L_1 + L_2C_2)P_x - L_2S_2P_y}{{P_x}^2 + {P_y}^2}$$

$$\theta_1 = tan^{-1}\frac{S_1}{C_1} = tan^{-1}\frac{(L_1 + L_2C_2)P_y - L_2S_2P_x}{(L_1 + L_2C_2)P_x + L_2S_2P_y}$$

Ecuación 25. Del elemento 3,4 de la ecuación 12 y 13

$$d_3 = -P_z - d_4$$

Ecuación 26. Obtención de Θ_3

$$\theta_3 = 0$$

Ecuación 27. Del elemento 1,1 de la ecuación 11 y 16 se obtiene C_4

$$C_4 = -n_xC_{12} + n_yS_{12}$$

Ecuación 28. Del elemento 2,1 de la ecuación 11 y 16 se obtiene S_4

$$S_4 = -n_xS_{12} + n_yC_{12}$$

Ecuación 29. Obtención de Θ_4

$$\theta_4 = tan^{-1}\frac{-n_x \sin(\theta_1 + \theta_2) + n_y \cos(\theta_1 + \theta_2)}{n_x \cos(\theta_1 + \theta_2) + n_y \sin(\theta_1 + \theta_2)}$$

2.10. LabVIEW

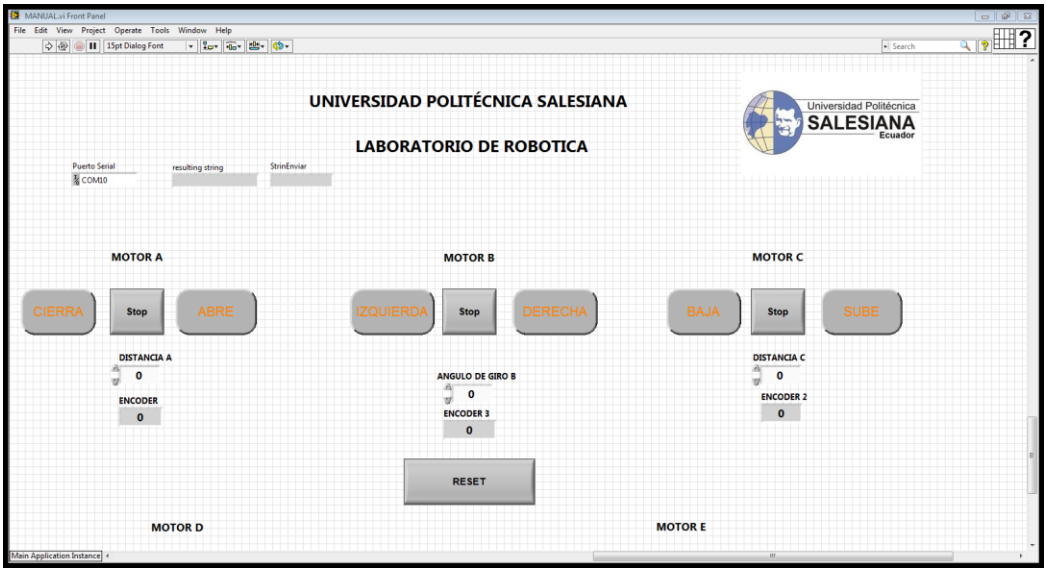
2.10.1. ¿Qué es LabVIEW?

(Acrónimo de Laboratory Virtual Instrumentation Engineering Workbench) es un lenguaje de programación gráfico diseñado para ingenieros y científicos para desarrollar aplicaciones de pruebas, control y medidas. (estuelectronic, 2012)

2.10.2. Panel frontal.

Es la interfaz con el usuario, se la utiliza para interactuar con el usuario cuando el programa se está ejecutando. En esta interfaz se definen los controles (se los usa como entradas, pueden ser botones, marcadores etc.) e indicadores (se los usa como salidas, pueden ser gráficas, etc). (Estuelectronic, 2013)

Figura 17. Panel frontal de LabVIEW

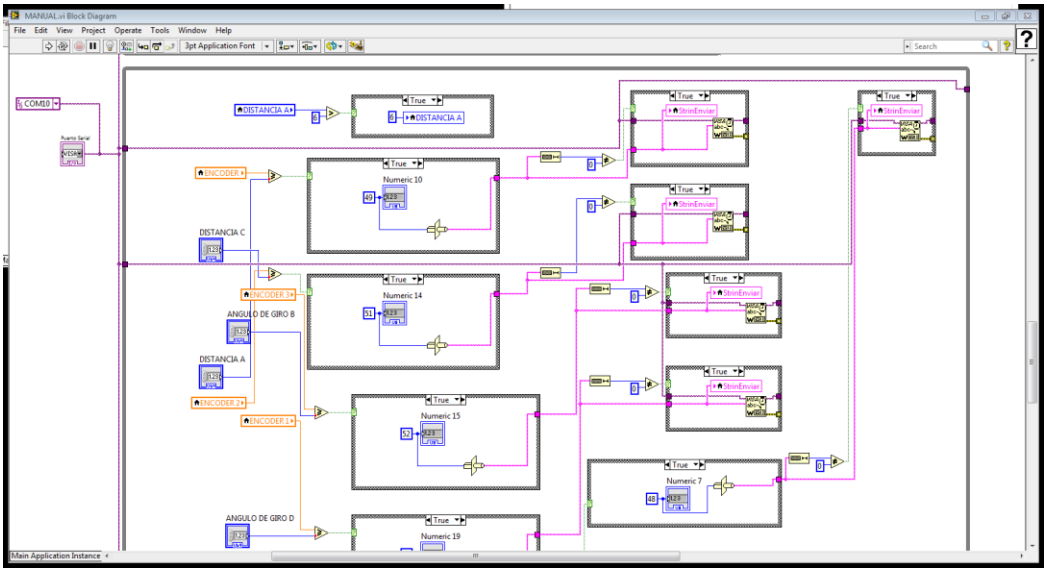


Elaborado por: Alejandra Ortiz & Edison Ponce

2.10.3. Diagrama de bloques.

Es el programa propiamente dicho, donde se define su funcionalidad, aquí se colocan íconos que realizan una determinada función y se interconectan (el código que controla el programa). Suele haber una tercera parte icono/conector que son los medios utilizados para conectar un VI con otros VIs. (Estuelectronic, 2013)

Figura 18. Diagrama de bloques de LabVIEW



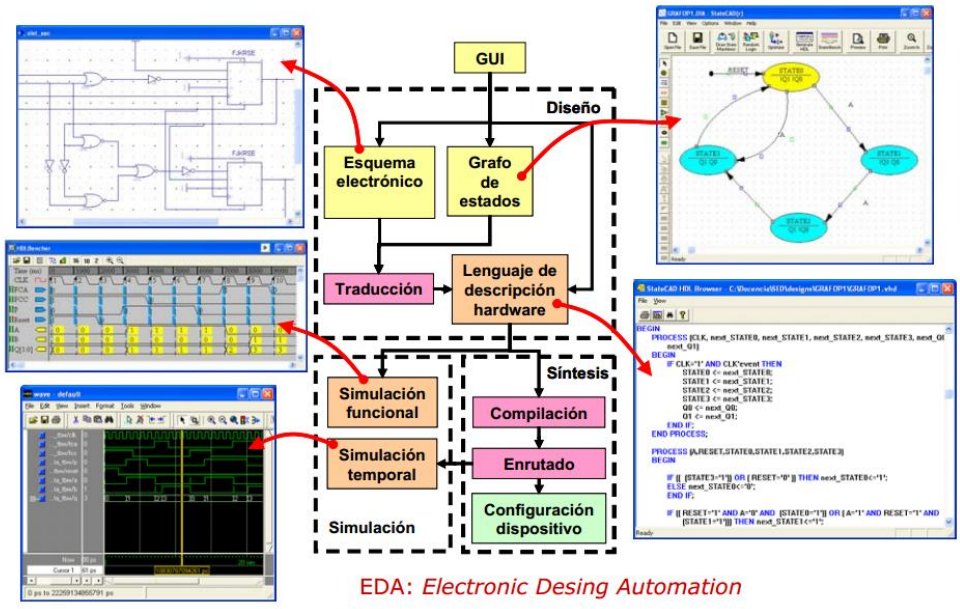
Elaborado por: Alejandra Ortiz & Edison Ponce

2.11. Project Navigator

Actualmente cualquier proceso de ingeniería dispone de un software que asiste al ingeniero con aplicaciones para el desarrollo de sistemas complejos. Los sistemas electrónicos reconfigurables del tipo FPGA son un buen ejemplo de la complejidad que se puede alcanzar, esta complejidad no sería abarcable sin la ayuda de un entorno con herramientas que asistan en el proceso de diseño, simulación, síntesis del resultado y configuración del hardware. Un ejemplo de un entorno de este tipo es el software de la empresa Xilinx denominado ISE (Integrated Software Environment). (Domínguez, 2011, pág. 3)

Este software constituye un verdadero entorno EDA (Electronic Desing Automation). La interfaz gráfica de usuario (GUI: Graphic User Interface) se denomina Proyect Navigator y facilita el acceso a todos los componentes del proyecto. Los diseños de usuario se pueden introducir mediante diferentes formatos. Los más utilizados son: los esquemáticos, los grafos de estados y las descripciones hardware en VHDL. Una vez compilados los diseños se puede simular su comportamiento a nivel funcional o a nivel temporal. A nivel funcional no tiene en cuenta los retardos provocados por el hardware y a nivel temporal simula el diseño teniendo en cuenta cómo se va a configurar el hardware. (Domínguez, 2011, pág. 3)

Figura 19. Esquema de los componentes más importantes del ISE



Fuente: (Domínguez, 2011, pág. 3)

2.12. Ajuste por mínimos cuadrados

Existen numerosas leyes físicas en las que se sabe de antemano que dos magnitudes "x" e "y" se relacionan a través de una ecuación lineal.

Ecuación 30. Ecuación de la recta.

$$y = ax + b$$

Donde las constantes "b" (ordenada en el origen) y "a" (pendiente) dependen del tipo de sistema que se estudia y, a menudo, son los parámetros que se pretende encontrar.

El método más efectivo para determinar los parámetros a y b se conoce como técnica de mínimos cuadrados.

Consiste en someter el sistema a diferentes condiciones, fijando para ello distintos valores de la variable independiente "x", anotando en cada caso el correspondiente valor medido para la variable dependiente "y". De este modo se dispone de una serie de puntos (x_1, y_1),.... (x_n, y_n) que, representados gráficamente, deberían caer sobre una línea recta. Sin embargo, los errores experimentales siempre presentes hacen que no se hallen perfectamente alineados. El método de mínimos cuadrados determina los valores de los parámetros "a" y "b" de la recta que mejor se ajusta a los datos experimentales. Sin detallar el procedimiento, se obtiene los parámetros de "a" y "b". (Torrelavega, 2008).

Ecuación 31. Obtención del parámetro "a"

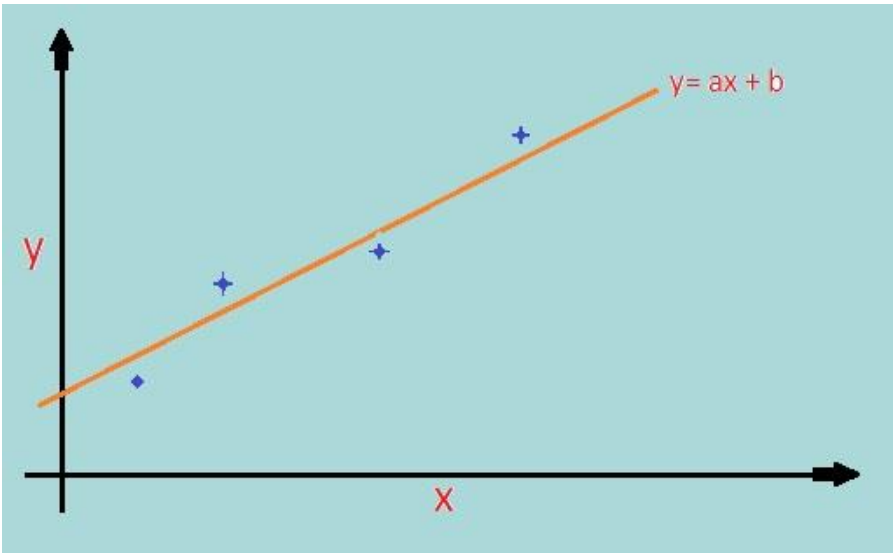
$$a = \frac{n\,(\sum x_i y_i) - (\sum x_i)(\sum y_i)}{n(\sum x_i^2) - (\sum x_i)^2}$$

Ecuación 32. Obtención del parámetro "b"

$$b = \frac{(\sum y_i) - a(\sum x_i)}{n}$$

Donde “n” es el número de medidas, “Σ” representa la suma de todos los datos que se indican e “i” representa el punto en el plano cartesiano.

Figura 20. Mínimos cuadrados



Elaborado por: Alejandra Ortiz & Edison Ponce

CAPÍTULO 3
DESARROLLO DE HARDWARE Y SOFTWARE

En este capítulo se desarrolla el diseño e implementación del hardware y software, el sistema de control está basado en una tarjeta FPGA Spartan 3E Starter. Además consta de tres tarjetas de potencia las mismas que usan un driver con el circuito integrado L298N. Los motores utilizados son tipo brushless.

La interfaz gráfica a utilizarse es LabVIEW con una comunicación serial entre la tarjeta Spartan 3E y la PC.

3.1. Diseño de hardware

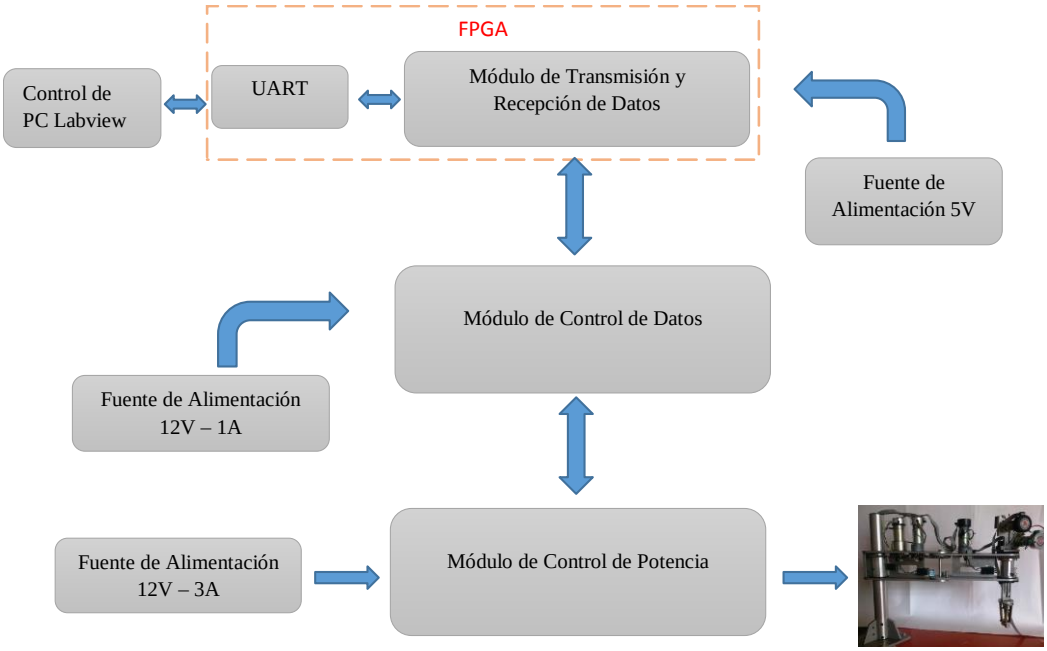
Se analiza el diagrama de bloques del sistema del brazo robótico RHINO Scara, para describir el funcionamiento de cada una de las fases que conforman y permiten su funcionabilidad.

3.1.1. Diagrama de bloques del sistema.

Las fases que forman parte del diagrama de bloques del hardware del brazo robótico Scara son:

- Fuentes de alimentación continúa.
- Tarjeta Spartan 3E para el control de movimiento del brazo y control de la comunicación entre ésta y la PC.
- Motores brushless con caja reductora y un enconder para el control del recorrido de los movimientos del brazo.
- Una tarjeta de control para envío y recepción de datos lógicos.
- Tarjetas de potencia para el control de los motores implementados con un circuito integrado L298N.
- Comunicación serial y computador para el proceso de información.

Figura 21. Diagrama de bloques del sistema de control del brazo Scara



Elaborado por: Alejandra Ortiz & Edison Ponce

3.1.1.1. Módulo de control de potencia mediante el puente H L298N para motores.

El circuito de potencia fue diseñado según algunos parámetros como: la cantidad de corriente (Amperios) y voltaje (Voltios) que utilizaría cada motor para el movimiento de su correspondiente articulación.

El módulo de control de potencia es basado en el circuito integrado L298N, diseñado para manejar diversos tipos de cargas como motores de corriente continua, motores paso a paso unipolares o bipolares, solenoides, etc.

Consta de 4 amplificadores de potencia que soportan intensidades de corriente de 2 A, con picos de hasta 3 A, controlables independientemente, o como 2 puentes H. En la versión "P" los amplificadores están conectados en paralelo 2 a 2, de modo que se obtiene 2 canales de 3 A, o 1 puente H de 3 A en modo continuo (3,5 A de pico). La capacidad de control de potencia supera los 120 W en total (160 W en intervalos cortos).

Dispone de 2 salidas para medir la corriente de cada puente H, que pueden ser llevadas a un ADC. (Heredia, 2014, pág. 50)

Figura 22. Módulo de control de potencia

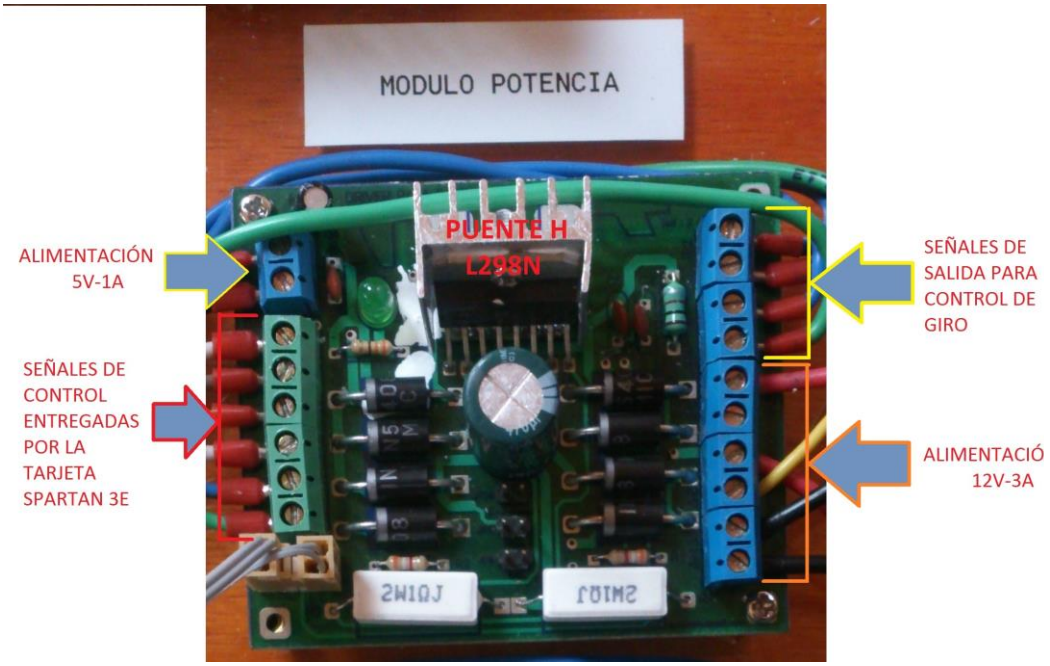


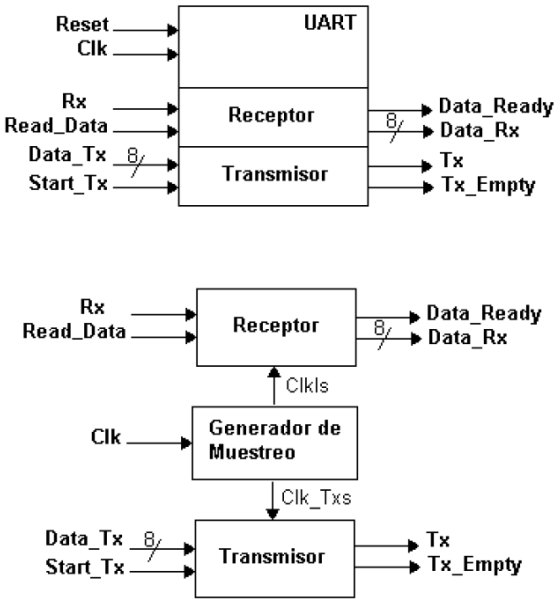
Imagen: Alejandra Ortiz & Edison Ponce

3.1.1.2. UART (Universal Asynchronous Receiver & Transmitter).

El Receptor y Transmisor Asíncrono Universal (UART) es el software indispensable de un sistema de comunicación serial. Su función consiste en convertir los datos que recibe de serie a paralelos y de paralelos a serie en el caso de los datos a enviar, es un proceso que se realiza bidireccionalmente pero no al mismo tiempo.

En las siguientes figuras se muestra el diagrama de bloques del UART y la tarjeta Spartan 3E con sus puertos seriales (macho y hembra).

Figura 23. Diagrama de bloques del UART



Fuente: (Bareño, 2013, pág. 76)

Figura 24. Ubicación del puerto serial

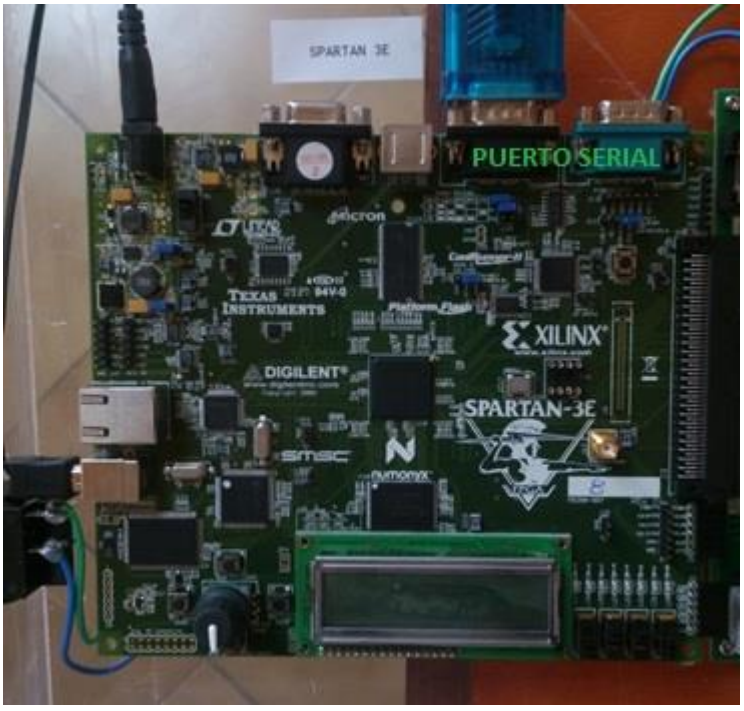


Imagen: Alejandra Ortiz & Edison Ponce

3.1.1.3. Módulo de control de datos.

Este módulo está constituido por tres circuitos: fuente de alimentación, tarjeta de acoplamiento de pines con la tarjeta Spartan 3E y la tarjeta Spartan 3E.

La fuente de alimentación consta de un regulador de tensión con una corriente máxima de 1 A a 12V.

La tarjeta de control consta de un regulador de voltaje LM7805 que permite la obtención de 5V. Sin embargo se necesita emplear un buen disipador de calor debido al comportamiento de este componente.

La tarjeta de acoplamiento permite unir los diferentes pines de la tarjeta de control con la tarjeta Spartan 3E para obtener la comunicación entre dichas tarjetas.

A continuación se detalla los pines de la tarjeta de acoplamiento.

Tabla 2. *Distribución de pines tarjeta de acoplamiento*

Motor		**A**	**B**	**C**	**D**	**E**
Encoder	Spartan	PIN 40	PIN 30	PIN 16	PIN 11	PIN 6
		PIN 42	PIN 32	PIN 26	PIN 13	PIN 8
	AVR	PIN 10	PIN 12	PIN 14	PIN 49	PIN 48
		PIN 4	PIN 11	PIN 13	PIN 51	PIN 50
Enable	Spartan	PIN 39	PIN 35	PIN 29	PIN 44	PIN 36
	AVR	PIN 17	PIN 7	PIN 15	PIN 5	PIN 6
Control	Spartan	PIN 37	PIN 33	PIN 27	PIN 14	PIN 9
		PIN 38	PIN 34	PIN 28	PIN 15	PIN 10
	AVR	PIN 37	PIN 40	PIN 42	PIN 45	PIN 47
		PIN 38	PIN 39	PIN 41	PIN 44	PIN 46
Reset	Spartan	PIN 25				
	AVR	PIN 20				
Led_inicio	Spartan	PIN 24				
	AVR	PIN 43				

Elaborado por: Alejandra Ortiz & Edison Ponce

Figura 25. Módulo de control de datos

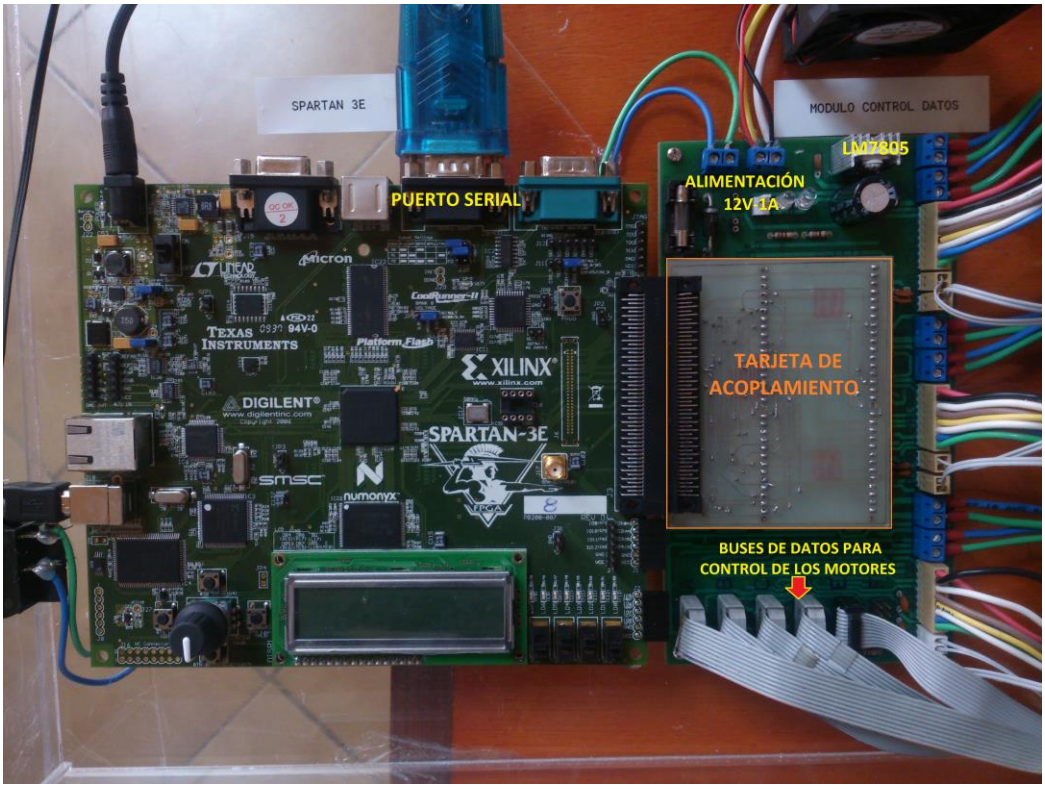


Imagen: Alejandra Ortiz & Edison Ponce

3.2. Desarrollo del software

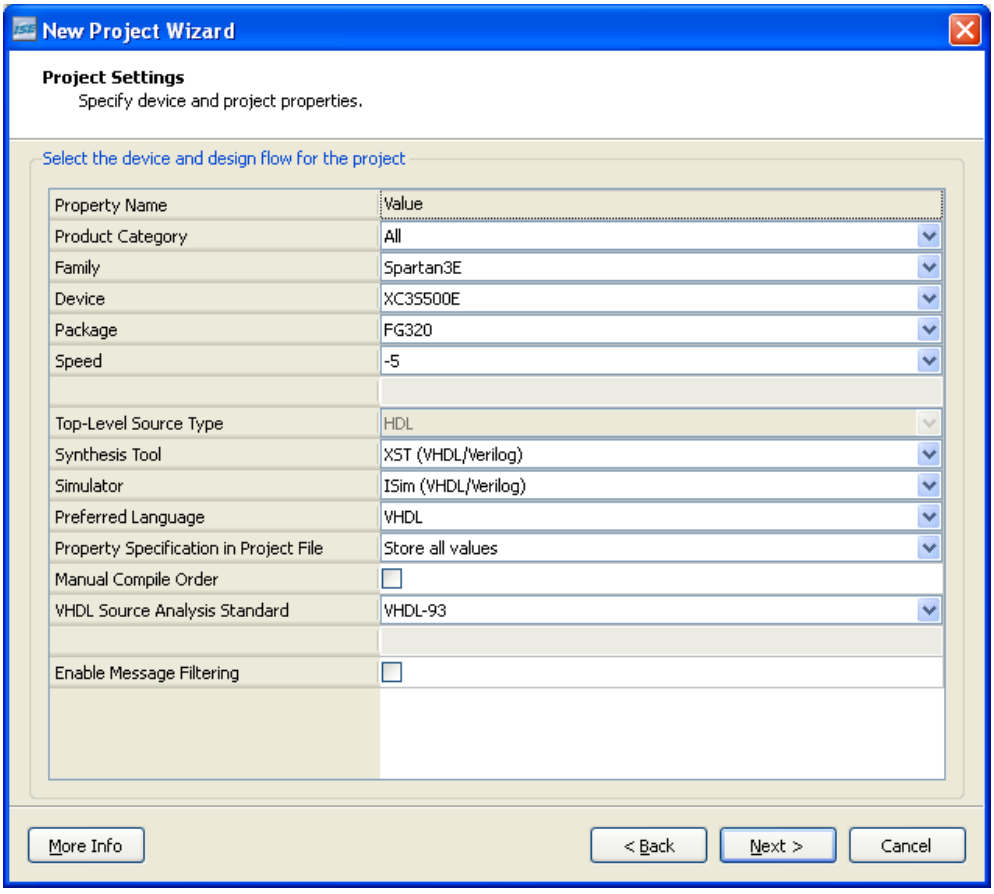
Se analiza la configuración básica de la tarjeta Spartan 3E para crear un nuevo proyecto en el software Project Navigator y los diagramas de flujo de las 3 etapas que forman parte del robot Scara tales como: los diagramas de flujo de la interfaz LabVIEW, los diagramas de flujo de la comunicación serial y los diagramas de flujo correspondiente al control del movimiento de los motores, como se muestra desde la página 43.

3.2.1. Configuración básica de la tarjeta Spartan 3E en Project Navigator.

Los pasos a seguir a continuación sirven para crear un nuevo proyecto y fijar los parámetros y propiedades de la tarjeta Spartan 3E, permitiendo un acople entre el software Project Navigator y el hadware correspondiente a la tarjeta.

1. Abra el Xilinx ISE Project Navigator, seleccionándolo desde el menú Inicio o desde el icono del escritorio.
2. En el Project Navigator, seleccione File, Menu, New Project. Nombre del proyecto "tutorial2". Para poner el proyecto en una ubicación diferente, directamente escriba la dirección del archivo en Project Location o busque haciendo clic en el botón Examinar de tres puntos.
3. La ubicación del proyecto siempre se puede ver en la parte superior de la pantalla del Project Navigator. Evite espacios en el nombre del proyecto o dirección. Haga clic en Siguiente y aparecerá la ventana de las propiedades del dispositivo.
4. Configurar las propiedades del dispositivo tal y como se muestra en la siguiente figura, a continuación seleccione Next hasta que se genere el proyecto.

Figura 26. Configuración de las propiedades del dispositivo



Elaborado por: Alejandra Ortiz & Edison Ponce

En la siguiente tabla se identifica las variables tipo strings utilizados en la programación y su respectivo código ASCII necesarios para este trabajo.

Tabla 3. *Strings utilizados en la programación y su respectivo código ASCII*

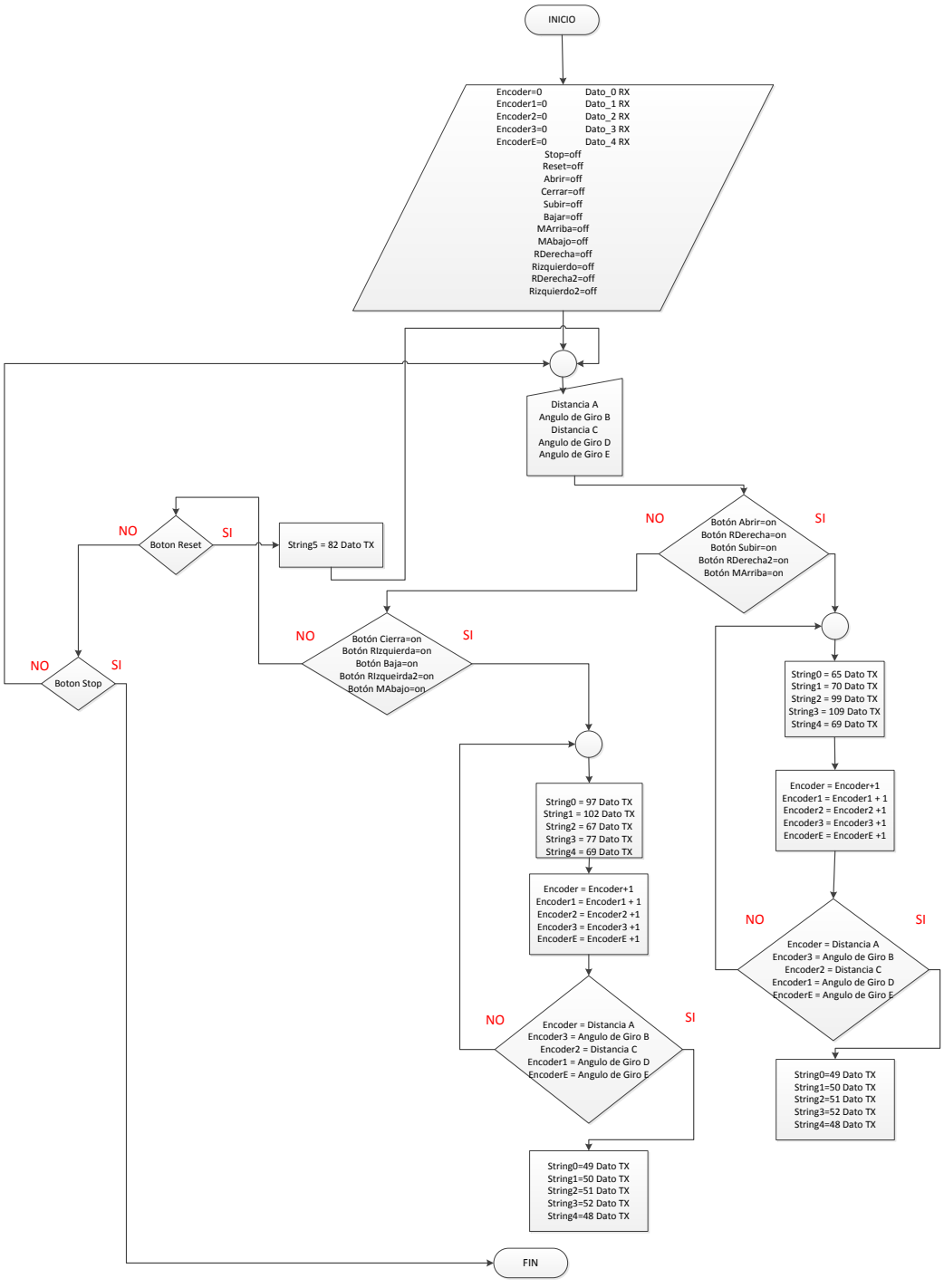
	Movimiento del motor		**Paro**	
Motor	**Letra**	**ASCII**	**Letra**	**ASCII**
A	A (Derecha)	65	1	49
	a (Izquierda)	97		
B	M (Derecha)	77	2	50
	m (Izquierda)	109		
C	C (Sube)	67	3	51
	c (Baja)	99		
D	F (Derecha)	70	4	52
	f (Izquierda)	102		
E	E (Derecha)	69	0	48
	e (Izquierda)	101		
Reset	R	82		
	r	114		

Elaborado por: Alejandra Ortiz & Edison Ponce

3.2.2. Diagrama de flujo para interfaz LabVIEW modo manual.

En la figura 27 se muestra el diagrama de flujo desarrollado en LabVIEW para la interfaz y envío de datos para el control de movimiento del robot Scara en modo manual.

Figura 27. Diagrama de flujo interfaz LabVIEW modo manual



Elaborado por: Aleiandra Ortiz & Edison Ponce

1. Se inicia el algoritmo empezando por declarar los datos, señales y variables con un valor por defecto igual a 0.

2. Se crea cinco variables tipo string correspondiente a los grados o distancia de cada motor las cuales se ingresarán manualmente.

3. Si el estado de las señales (Abrir, RDerecha, Subir, RDerecha2, MArriba) cambian a ON se procede a enviar los datos del siguiente bloque por el pin de transmisión Serial Tx correspondiente al movimiento del motor, caso contrario si el estado es OFF se procede a escoger las otras cinco señales posibles de los motores respectivamente (Cierra, RIzquierda, Baja, RIzquierda2, MAbajo).

4. Este bloque contiene los cinco diferentes datos a enviar por el pin de transmisión serial Tx correspondiente al movimiento de cada motor (String0, String1, String2, String3, String4).

5. Se realiza un contador para el encoder de los motores (Encoder, Encoder1, Encoder2, Encoder3, EncoderE), los datos de los encoders son recibidos mediante el pin de recepción serial RX.

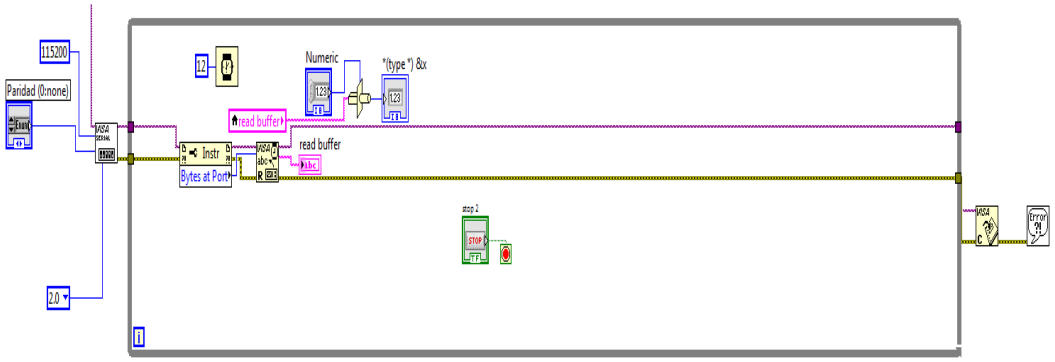
6. Si el contador del encoder es igual a la variable tipo string introducida manualmente, correspondiente a los grados o distancias de cada motor, se enviara el dato correspondiente al paro del motor por el pin de transmisión serial Tx.

7. Si el estado de la señal "RESET" cambia a ON se enviara el dato correspondiente para ubicar el brazo robótico en posición inicial.

8. Si el botón Stop de LabVIEW es activado, el proceso finaliza.

Se observa la configuración del puerto de la comunicación serial que existe entre la tarjeta FPGA Spartan 3E y la PC, valores tales como: velocidad de transmisión de 115200 bauds/seg, 2 bits de parada, paridad ninguna y un retardo de 12 mseg en este proceso.

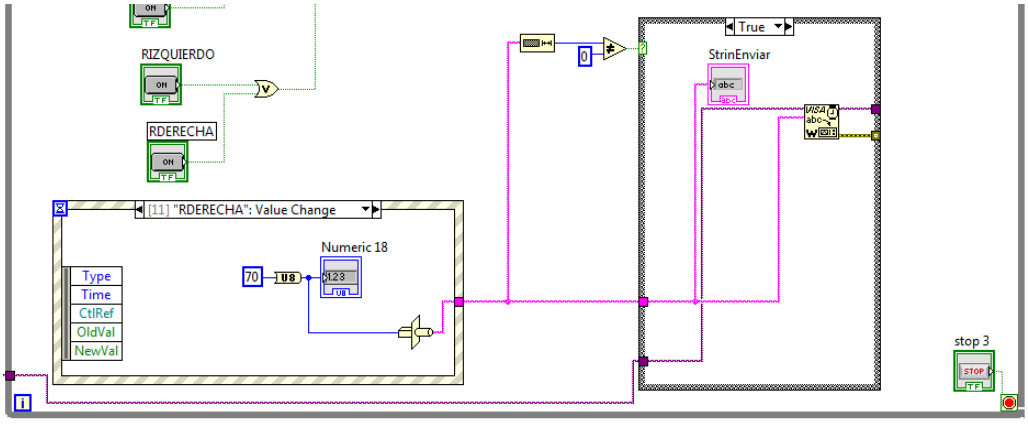
Figura 28. Configuración de la comunicación serial en LabVIEW



Elaborado por: Alejandra Ortiz & Edison Ponce

El número ASCII, correspondiente a cada movimiento, es transformado a un intenger de 8 bits sin signo, para luego ser enviado por el puerto serial de la PC hacia la tarjeta FPGA.

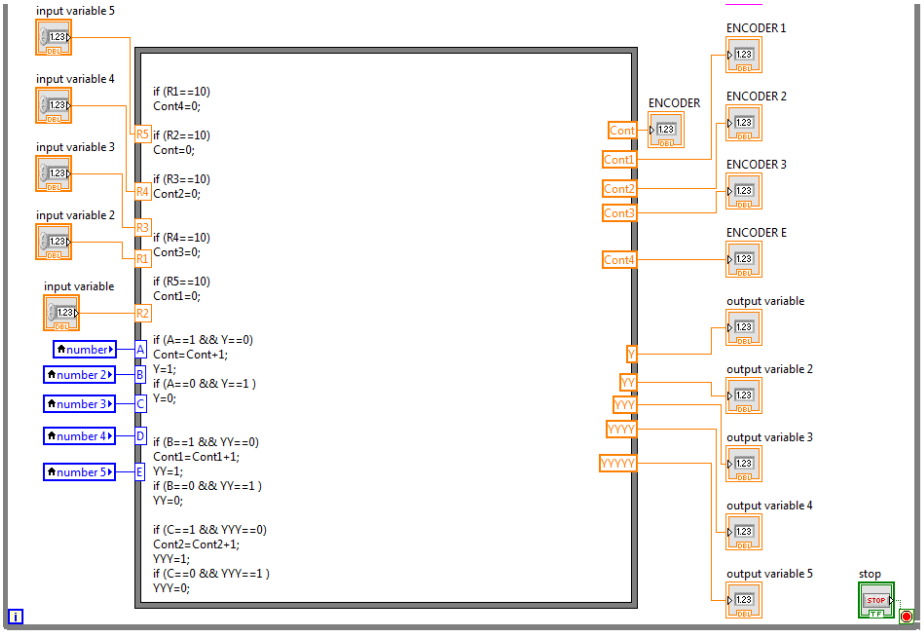
Figura 29. Envío del número integer de 8 bits sin signo correspondiente al movimiento



Elaborado por: Alejandra Ortiz & Edison Ponce

Se implementa un contador para el cambio de estado de cada bit del byte recibido desde la tarjeta FPGA correspondiente al movimiento del motor, con la finalidad de tener un control en el número de vueltas que realiza el encoder.

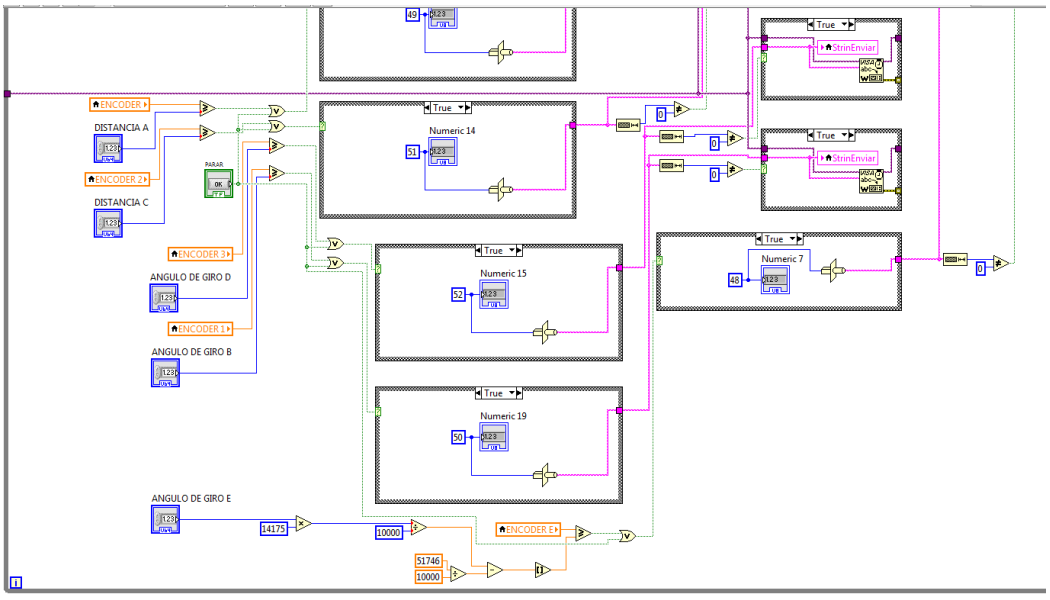
Figura 30. Contadores para el cambio de estado



Elaborado por: Alejandra Ortiz & Edison Ponce

El número ASCII, correspondiente a cada paro, es transformado a un intenger de 8 bits sin signo, para luego ser enviado por el puerto serial de la PC hacia la tarjeta FPGA.

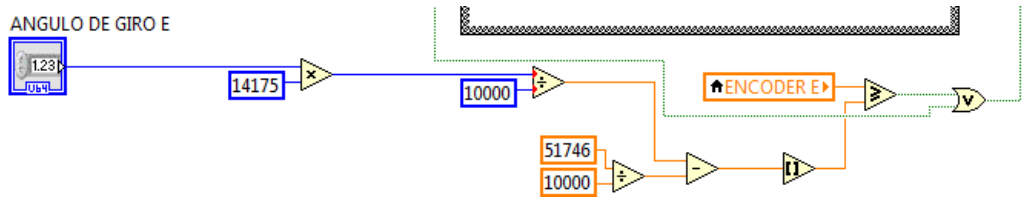
Figura 31. Envío del número integer de 8 bits sin signo correspondiente al paro



Elaborado por: Alejandra Ortiz & Edison Ponce

Se implementa la fórmula del ajuste de mínimo cuadrados con el objetivo de tener una linealidad en el movimiento.

Figura 32. Implementación de la fórmula de ajustes de mínimo cuadrados

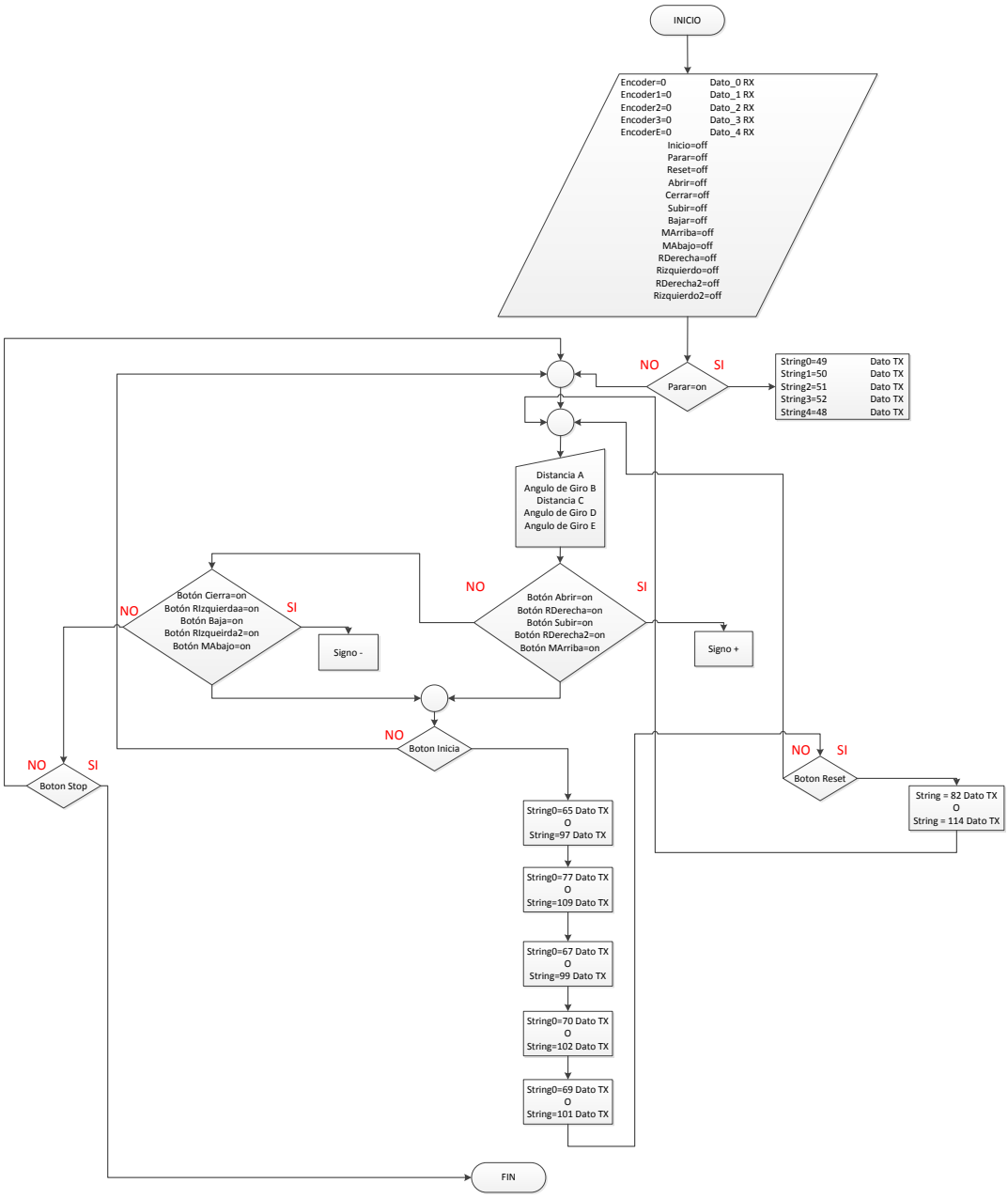


Elaborado por: Alejandra Ortiz & Edison Ponce

3.2.3. Diagrama de flujo para interfaz LabVIEW modo automático.

En la figura 33 se muestra el diagrama de flujo desarrollado en LabVIEW para la Interfaz y envió de datos para el control de movimiento del robot Scara en modo automático.

Figura 33. Diagrama de flujo interfaz modo automático



Elaborado por: Alejandra Ortiz & Edison Ponce

1. Se inicia el algoritmo empezando por declarar los datos, señales y variables con un valor por defecto igual a 0.

2. Si la variable string “Parar” cambia a ON se envía los datos correspondientes al paro de cada motor por el pin de transmisión serial TX.

3. Caso contrario se crea cinco variables tipo string correspondiente a los grados o distancia de cada motor las cuales se ingresarán manualmente.

4. Si el estado de las señales (Abrir, RDerecha, Subir, RDerecha2, MArriba) cambian a ON se procede asignar un string “+”.

5. Caso contrario si el estado de las señales (Cierra, RIzquierda, Baja, RIzquierda2, MAbajo) cambian a ON se procede asignar un string “-”.

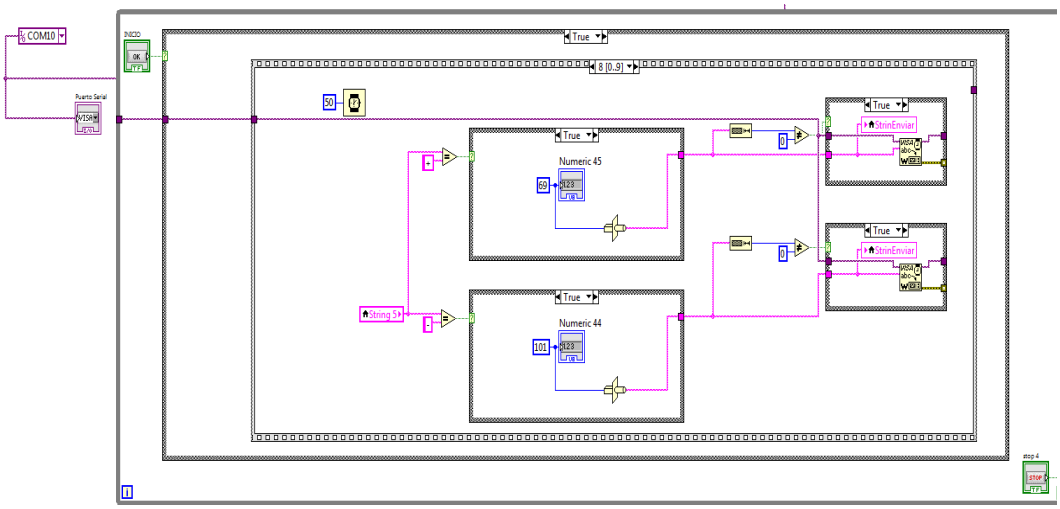
6. Si el estado de la señal “Inicia” cambia a ON, se procede a enviar el string correspondiente al movimiento del motor A, luego el motor B, motor C, motor D y motor E con un determinado intervalo de tiempo entre cada uno de éstos.

7. Si el estado de la señal “Reset” cambia a ON, se envía el dato para ubicar el brazo robótico en la posición inicial, sino el proceso vuelve a su inicio.

8. Si el botón Stop de LabVIEW es activado, el proceso finaliza.

El número ASCII, correspondiente a cada movimiento, es transformado a un intenger de 8 bits sin signo, este número se elige manualmente mediante el signo “+ o -” y se lo ejecuta en un ciclo previamente configurado con sus respectivos tiempos de retardo entre éstos, para luego ser enviado por el puerto serial hacia la tarjeta FPGA.

Figura 34. Envío del número integer de 8 bits sin signo en un ciclo de eventos

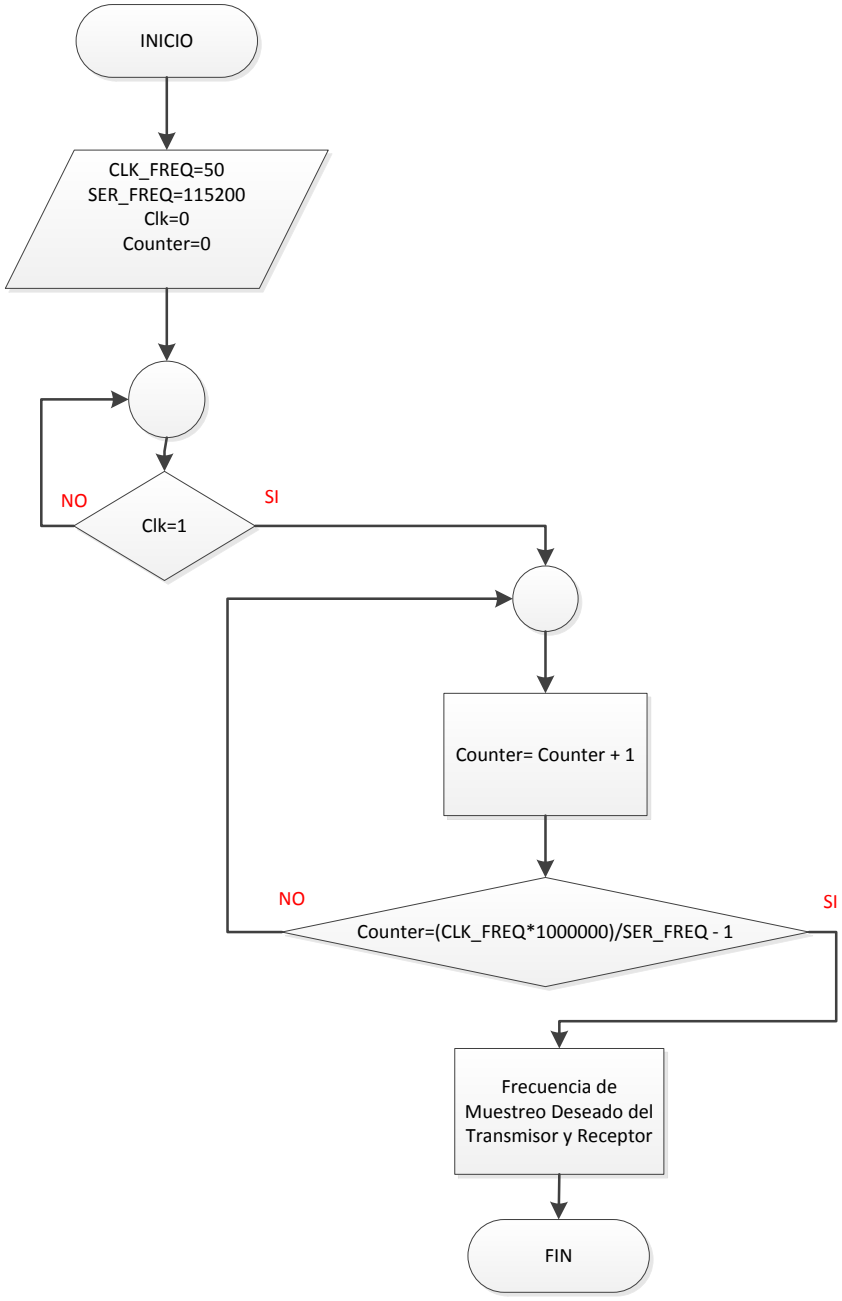


Elaborado por: Alejandra Ortiz & Edison Ponce

3.2.4. Diagrama de flujo divisor de frecuencia.

En la figura 35 se muestra el diagrama de flujo correspondiente a la obtención de la frecuencia deseada para la comunicación serial entre la tarjeta Spartan 3E y la PC.

Figura 35. Diagrama de flujo frecuencia de muestreo del receptor y transmisor



Elaborado por: Alejandra Ortiz & Edison Ponce

1. Se inicia el algoritmo ingresando constantes, señales y variables. Las señales y variables se inicializan en 0, las constantes dependen de la frecuencia de trabajo de la tarjeta Spartan 3E, que es de 50 MHz y la velocidad de

transmisión deseada para la comunicación, dicha velocidad se eligió de entre las más comunes, ésta se estableció en 115200 bits/s.

2. Si la señal de reloj “clk” cambia a estado “1” se procede con el siguiente bloque, caso contrario se mantiene en espera de cambio de estado.

3. Se crea una variable “counter” para ser usada como un contador.

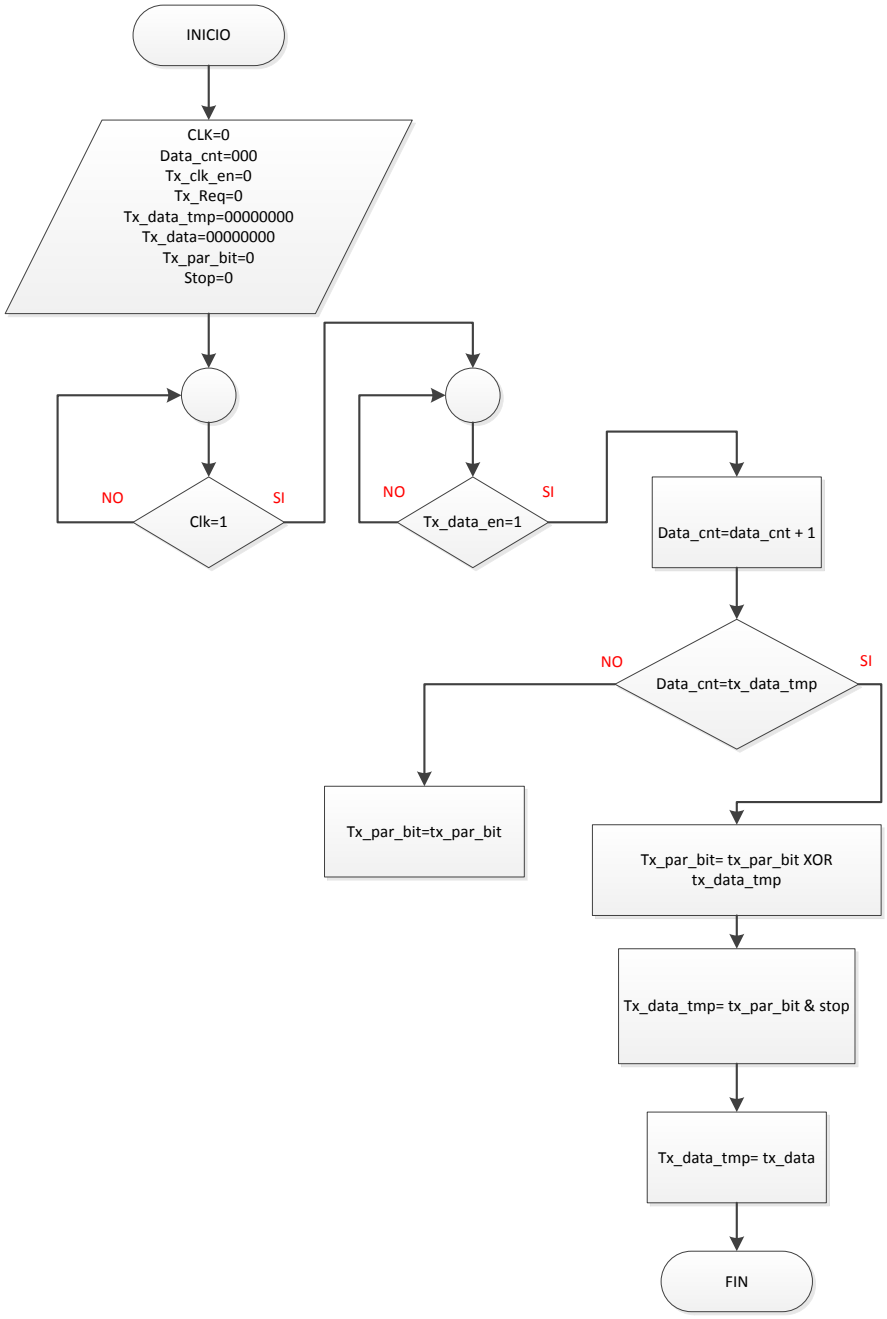
4. Si la variable “counter” es igual a la frecuencia de reloj de la tarjeta, dividido para la frecuencia deseada, dicho valor menos uno. Se obtiene el número de ciclos que el proceso se leerá, adquiriendo la frecuencia necesaria para utilizarla en los procesos de transmisión y recepción.

5. Se finaliza el proceso.

3.2.5. Diagrama de flujo trama de transmisión.

En la figura 36 se muestra el diagrama de flujo correspondiente a la obtención de la trama a enviarse por el pin de transmisión Serial Tx de la tarjeta Spartan 3E hacia la PC.

Figura 36. Diagrama de flujo trama de transmisión



Elaborado por: Alejandra Ortiz & Edison Ponce

1. Se inicializa el algoritmo con la declaración de datos tipo vector, señales y variables inicializados en 0.

2. Si la señal de reloj “clk” cambia a estado “1” continua al siguiente bloque, caso contrario se mantiene en espera del cambio de estado.

3. Si la señal “Tx_data_en” cambia a estado “1”, se implementará un contador con la variable “data_cnt”, caso contrario se mantiene en espera del cambio de estado.

4. Si la variable “data_cnt” es igual a la señal “tx_data_tmp”, se efectuará la unión de la señal “tx_par_bit” con la señal “tx_data_tmp” almacenada en la misma señal “tx_par_bit”, caso contrario la señal “tx_par_bit” se mantiene en su estado inicial.

5. En este bloque se crea la señal “tx_par_bit” más la señal “stop” ingresada en la señal “tx_data_tmp”.

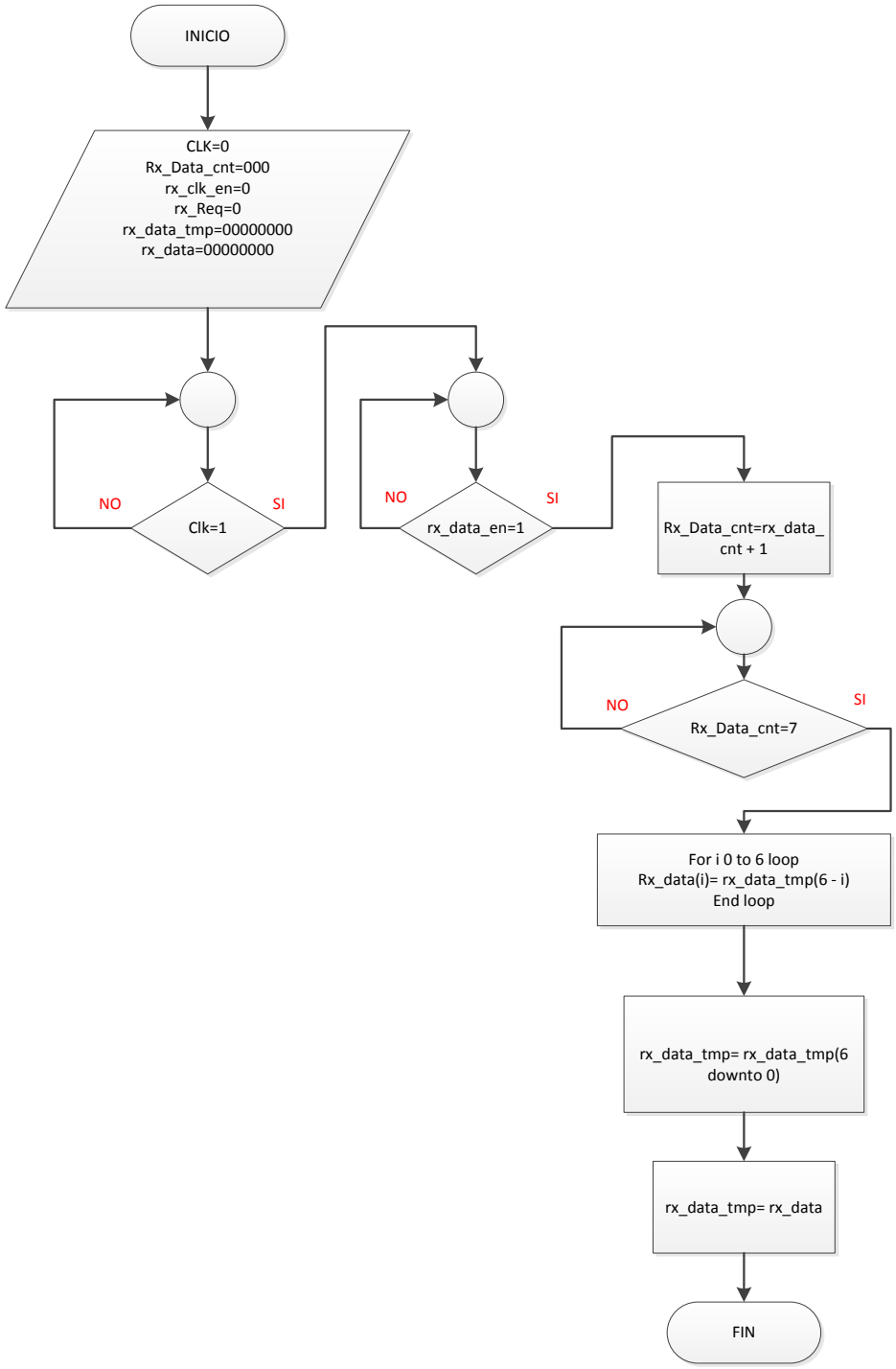
6. Finalmente se iguala la señal “tx_data_tmp” al dato “tx_data”. Este es enviado por el pin de transmisión serial Tx de la tarjeta.

7. Finaliza el proceso.

3.2.6. Diagrama de flujo trama de recepción.

En la figura 37 se muestra el diagrama de flujo correspondiente a la verificación de la trama a recibirse por el pin de recepción Serial Rx de la tarjeta Spartan 3E desde la PC.

Figura 37. Diagrama de flujo trama recepción



Elaborado por: Alejandra Ortiz & Edison Ponce

1. Se inicializa el algoritmo con la declaración de datos tipo vector, señales y variables inicializados en cero.

2. Si la señal de reloj “clk” cambia a estado “1” continua al siguiente bloque, caso contrario se mantiene en espera del cambio de estado.

3. Si la señal “rx_data_en” cambia a estado “1”, se implementará un contador con la variable “rx_data_cnt”, caso contrario se mantiene en espera del cambio de estado.

4. Si “rx_data_cnt” es igual a 7, se implementará un lazo for de siete ciclos que mostrará el dato “rx_data_tmp” que se encuentra en las siete diferentes posiciones, caso contrario se esperará hasta que “rx_data_cnt” sea igual a 7.

5. Se crea la señal “rx_data_tmp” de siete posiciones.

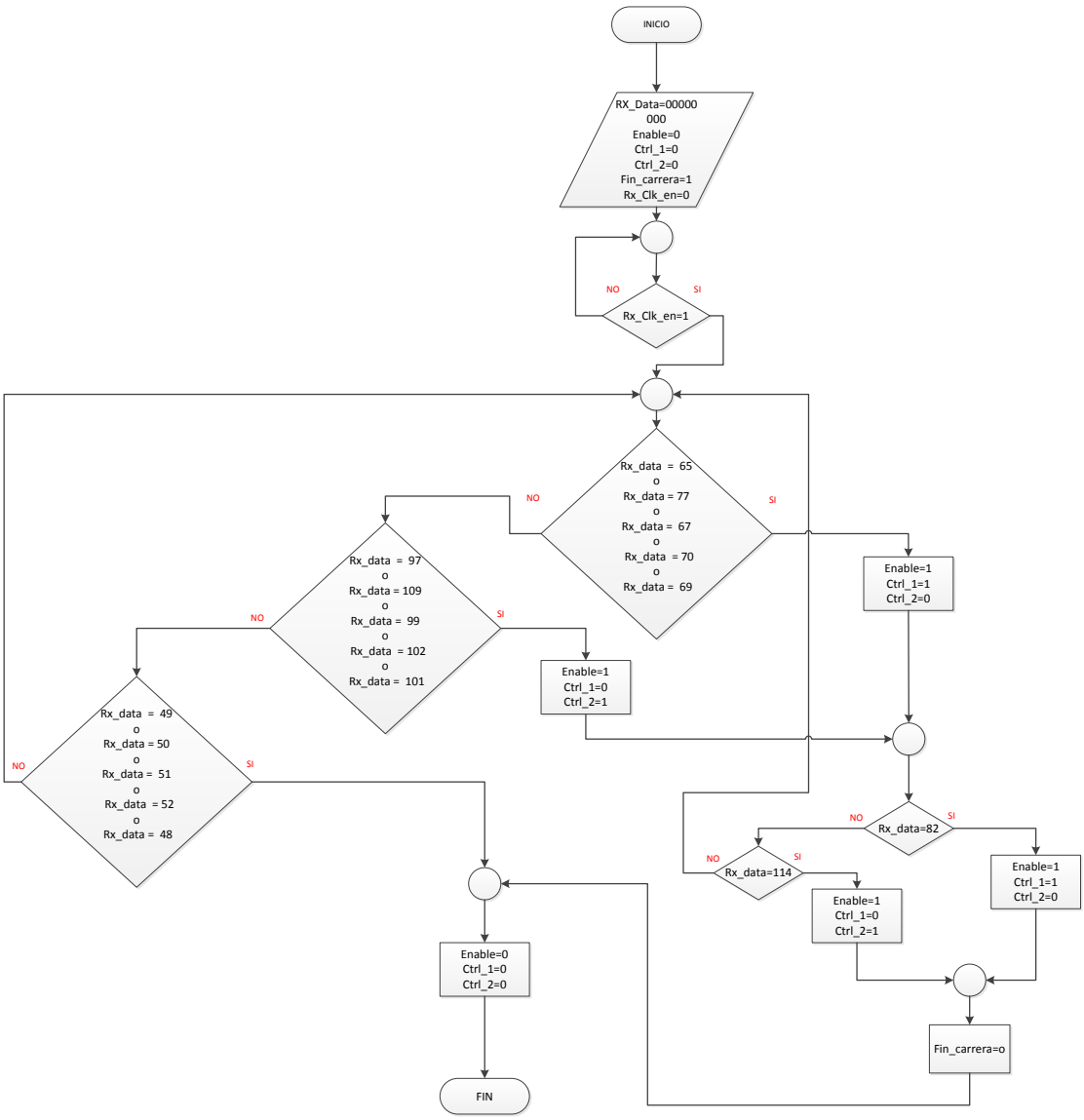
6. Finalmente se iguala la señal “rx_data_tmp” al dato “rx_data”. Este es recibido por el pin de recepción serial Rx de la tarjeta.

7. Finaliza el proceso.

3.2.7. Diagrama de flujo movimiento de motores proceso de recepción.

En la figura 38 se muestra el diagrama de flujo correspondiente a la recepción de la trama enviada desde LabVIEW hacia la tarjeta Spartan 3E.

Figura 38. Diagrama de flujo movimiento de motores proceso de recepción



Elaborado por: Alejandra Ortiz & Edison Ponce

1. Inicia el proceso declarando los datos tipo vector, señales y variables inicializadas en 0, exceptuando la señal “fin_carrera=1”.

2. Si el estado de la señal “Rx_clk_en” cambia a 1 continua con el siguiente bloque, caso contrario se mantiene en espera del cambio de estado.

3. Si el dato recibido “Rx_data” es igual a “65 ó 77 ó 67 ó 70 ó 69” se declara las variables “enable=1, Ctrl_1=1, Ctrl_2=0” que permiten el movimiento del

motor, caso contrario se recibirá el dato “Rx_data” igual a “97 ó 109 ó 99 ó 102 ó 101”, que permite el cambio de las variables “enable=1, Ctrl_1=0, Ctrl_2=1” con un movimiento inverso al anterior. Ver tabla 2.

4. Si el dato recibido “Rx_data” es igual a “49 ó 50 ó 51 ó 52 ó 48” se declara las variables “enable=0, Ctrl_1=0, Ctrl_2=0” que permiten el paro del motor, caso contrario. Ver tabla 2.

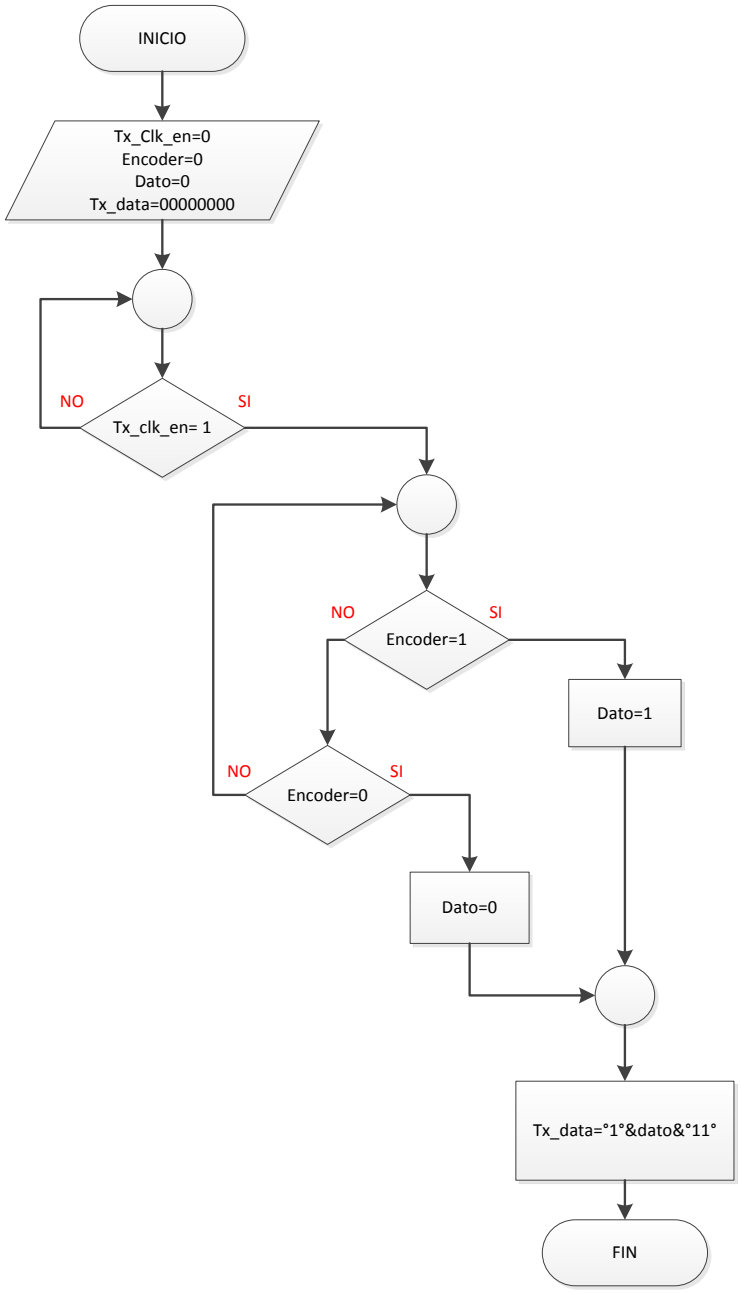
5. Una vez realizado el movimiento del brazo, si el dato recibido “rx_data” es igual a “82 ó 114”, dicho brazo se reinicia a su posición inicial cambiando la señal “fin_carrera” igual a 0’. Ver tabla 2.

6. Finalización del proceso.

3.2.8. Diagrama de flujo movimiento de motores proceso de transmisión.

En la figura 39 se muestra el diagrama de flujo correspondiente a la transmisión de la trama recibida por LabVIEW desde la tarjeta Spartan 3E.

Figura 39. Diagrama de flujo movimiento de motores proceso de recepción



Elaborado por: Alejandra Ortiz & Edison Ponce

1. Inicia el proceso declarando los datos tipo vector, señales y variables inicializadas en 0.

2. Si el estado de la señal "Tx_clk_en" cambia a 1 continua con el siguiente bloque, caso contrario se mantiene en espera del cambio de estado.

3. Si la señal del “encoder” cambia de estado a 1 el dato “dato=1”.

4. Caso contrario si el estado del “encoder” cambia de estado a 0 el dato “dato=0”, caso contrario se mantiene en espera del cambio de estado.

5. Se crea el dato (Tx_data = “1” + “dato” + “11”), completando así la trama de un byte a enviarse por el pin de transmisión serial Tx.

6. Finalización del proceso.

CAPÍTULO 4
ANÁLISIS DE COSTOS

En este capítulo se analiza los costos de: diseño y construcción del hardware, desarrollo del software y el costo total que conllevó el proyecto de investigación.

4.1. Costos de hardware

La tabla 4 señala los costos de los materiales que se utilizaron en la construcción del brazo robótico, tanto la parte electrónica y ensamblaje del equipo.

Tabla 4. *Costos de hardware*

Cantidad	Detalle	Valor Unitario	Total
1	Tarjeta FPGA Spartan 3E	150	150
3	Placas de Control	80	240
1	Tarjeta Expansion de Pines	25	25
1	Estructura de Acrilico	56	56
4	LM393N	2,50	10,00
1	Fuente de Alimentacion AC/DC 1A	50	50,00
3	L298N	4	12,00
24	1N5408	0,60	14,40
6	Resistencias 2W/ 1Ω	0,40	2,40
3	Capacitor 470µf/ 35V	0,80	2,40
12	Capacitor 0,1µf	0,10	1,20
3	Resitencia 330Ω	0,10	0,30
3	Inductancia	0,35	1,05
18	Header 2 entradas	0,20	3,60
6	Header 3 entradas	0,30	1,80
3	Resistencias 3,9K	0,10	0,30
3	Diodo Led	0,10	0,30
3	Disipador de Corriente	1,50	4,50
3	Cable #20	0,20	0,60
2	Cable Bus de Datos	0,60	1,20
	TOTAL		$ 577

Elaborado por: Alejandra Ortiz & Edison Ponce

4.2. Costo de diseño de hardware

A continuación se detalla el costo de diseño del circuito electrónico.

Tabla 5 . *Costos de diseño de hardware*

Descripción	**Cantidad (personas)**	**Tiempo (horas)**	**Precio. Cada Hora. $**	**Precio. Total del Trabajo. $**
Diseño de ingeniería.	2	60	6	720
TOTAL				**720 $**

Elaborado por: Alejandra Ortiz & Edison Ponce

4.3. Costos de desarrollo del software

Los costos analizados para el desarrollo del software del robot Scara son: LabVIEW, Proyect Navigator (Lenguaje VHDL), especificando el valor por cada hora de servicios prestados, como se muestra en la tabla 6.

Tabla 6. *Costo de desarrollo de software*

Descripción	**Cantidad (personas)**	**Tiempo (horas)**	**Precio. Cada Hora. $**	**Precio. Total del Trabajo. $**
Programación LabVIEW	2	80	5$	800$
Programación VHDL	2	200	5$	2000
TOTAL				**2800 $**

Elaborado por: Alejandra Ortiz & Edison Ponce

4.4. Costo total del proyecto

A continuación se detalla el precio total para el diseño y construcción del robot Scara, en la tabla 7 se especifican los costos del hardware y desarrollo del software.

Tabla 7. *Costo total del proyecto*

DESCRIPCIÓN	***COSTO***
Hardware	577$
Diseño de hardware	720 $
Desarrollo de software	2800 $
TOTAL	**4097$**

Elaborado por: Alejandra Ortiz & Edison Ponce

CONCLUSIONES

- En el análisis e implementación de la cinemática directa se obtuvo un campo interactivo para cada motor del brazo, conformando un movimiento coherente y acorde a lo solicitado por el usuario, dentro de un error característico de estos motores del 3%, lo que se considera aceptable para los fines propuestos en este tema de investigación.

- Para construir el algoritmo UART con el código VHDL se logró que los procesos construidos en este algoritmo funcionen en paralelo, facilitando la selección requerida para cada acción, haciendo más eficiente el tiempo de respuesta en el movimiento del brazo robótico, ya que no se requiere la lectura de todo el algoritmo.

- Se realizaron las respectivas pruebas en la tarjeta, concluyendo que su campo didáctico entre la interacción del usuario y la Spartan 3E en las diferentes practicas prediseñadas tienen un nivel aceptable al momento de visualizar el funcionamiento de una tarea dedicada, con una velocidad de transmisión y recepción máxima de 115200 Mbps entre la tarjeta Spartan 3E y la interfaz LabVIEW, concluyendo también la necesidad de implementar paros de emergencia en el software para controlar una posible mala manipulación de dicho brazo.

- El nivel de seguridad y confiabilidad que se obtuvo se encuentra dentro de los márgenes de un estudio eficiente en la manipulación de un brazo robótico en el campo universitario, en base al movimiento de cada motor que conforma el brazo se ha logrado incluir actividades prediseñadas con sus diferentes modos de manipulación (manual, automático y vía bluetooth).

RECOMENDACIONES

- Se recomienda tener un gran cuidado en la polarización a tierra de la tarjeta con el área de control de datos, debido a que cada una posee diferentes niveles de voltaje para su correcto funcionamiento, adicionalmente la tarjeta consta de switchs que permiten el cambio de voltaje de 5v – 3.3v con lo que el no tener un mismo voltaje en las mismas áreas repercutiría en el funcionamiento y manipulación de los datos digitales.

- El tener una idea de los bloques o procesos a construir en el área del UART es de vital importancia, puesto que sin una idea clara se llega a confusiones y a una mala apreciación del trabajo que se requiere para la transmisión y recepción de datos, poniendo gran énfasis en construir un algoritmo que tenga la facilidad de cambiar sus variables a futuro de una forma sencilla y optima en el momento que se lo requiera.

- Tener un gran cuidado en la manipulación y conexión de los buses de datos que van desde los motores hacia la tarjeta de control es de gran importancia debido a que el encoder con el que trabaja el motor es muy susceptible a daños por una mala polarización, ya que en dichos buses se transmiten voltajes de 12v que son mucho mayores al voltaje de 5v necesario para su funcionamiento.

LISTA DE REFERENCIAS

Arian. (15 de noviembre de 2012). *Cochesrc.* Recuperado el 22 de septiembre de 2013, de Cochersc: http://www.cochesrc.com/motor-electrico-brushless-funcionamiento-y-caracteristicas-a3607.html

Bareño, C. I. (20 de octubre de 2013). *Universidad Nacional de Colombia_ Electronica Digital.* Recuperado el 9 de enero de 2014, de http://gmun.unal.edu.co/~cicamargoba/digital2/1.pdf

Boada, Y. F., & Morales, L. A. (16 de junio de 2010). *Escuela Politécnica Nacional.* Recuperado el 28 de julio de 2013, de Escuela Politécnica Nacional: http://bibdigital.epn.edu.ec/bitstream/15000/2219/1/CD-2976.pdf

CEM-bot. (22 de julio de 2013). *Wikipedia.* Recuperado el 20 de enero de 2014, de Wikipedia: http://es.wikipedia.org/wiki/Field_Programmable_Gate_Array

Domínguez, M. Á. (18 de Enero de 2011). *Universidad de Vigo.* Recuperado el 14 de noviembre de 2013, de http://webs.uvigo.es/mdgomez/SED/Guia_Inicio_ISE.pdf

estuelectronic. (13 de febrero de 2012). *estuelectronic*. Recuperado el 20 de agosto de 2013, de estuelectronic: http://estuelectronic.wordpress.com/2012/08/06/que-es-y-para-que-sirve-labview/

Estuelectronic. (20 de marzo de 2013). *Estuelectronic*. Recuperado el 13 de febrero de 2014, de Estuelectronic: http://estuelectronic.wordpress.com/2012/08/06/que-es-y-para-que-sirve-labview/

Heredia, J. (2014). *Diseño e implementación de dos controladores para los brazos robóticos: Antropomórfico y Scara, utilizando microcontroladores Atmel AVR'S.* Quito: Tesis de Pregrado.

Hernández, M. (1 de marzo de 2013). *Ingeniería de Sistemas y Automática*. Recuperado el 16 de abril de 2014, de Ingeniería de Sistemas y Automática: http://isa.umh.es/asignaturas/rvc/tema4.pdf

Inc, X. (20 de enero de 2011). *Xilinx.* Recuperado el 15 de juliio de 2013, de Xilinx: http://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf

Inc, X. (19 de julio de 2013). *Xilinx.* Recuperado el 5 de mayo de 2014, de Xilinx: http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf

Marín, J. M. (2013). *Biblioteca de Ingenieria (Universidad de Sevilla).* Recuperado el 10 de septiembre de 2013, de Biblioteca de Ingenieria (Universidad de Sevilla): http://bibing.us.es/proyectos/abreproy/11375/fichero/MEMORIA%252FFPGA%C2%B4S.pdf

Noriega, A., Muñiz, M., & García, A. (20 de Agosto de 2010). *xixcnim.* Recuperado el 16 de octubre de 2013, de xixcnim: http://www.xixcnim.uji.es/CDActas/Documentos/ComunicacionesOrales/14-02.pdf

Research, E. J. (9 de enero de 2012). *Academia.edu.* Recuperado el 9 de abril de 2014, de https://www.academia.edu/5134376/Kinematic_Modeling_and_Simulation_of_a_SCARA_Robot_by_Using_Solid_Dynamics_and_Verification_by_MATLAB_Simulink

rhinorobotics LTD. (25 de diciembre de 2013). Recuperado el 20 de febrero de 2014, de rhinorobotics LTD: http://www.rhinorobotics.com/scara.html

Torrelavega. (6 de agosto de 2008). *Universidad de Cantabria.* Recuperado el 26 de febrero de 2014, de Universidad de Cantabria: http://ocw.unican.es/ensenanzas-tecnicas/fisica-i/practicas-1/Ajuste%20por%20minimos%20cuadrados.

U-Cursos. (12 de Marzo de 2006). *U-Cursos.* Recuperado el 23 de julio de 2013, de U-Cursos: https://www.u-cursos.cl/ingenieria/2006/1/EL693/1/material.../86389

Xilinx. (2013). *Xilinx.* Recuperado el 09 de abril de 2014, de http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf

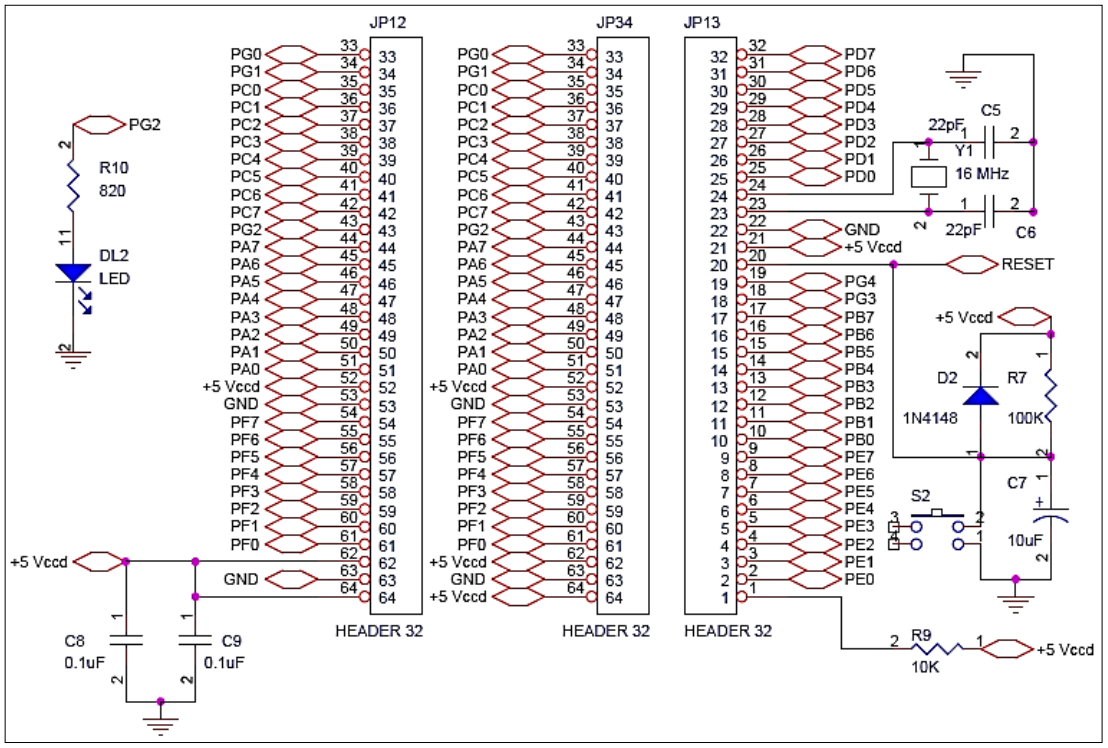
Anexo 1. Distribución de pines tarjeta Spartan 3E

Signal Name	FPGA Pin	Shared Header Connections					FX2 Connector		FPGA Pin	Signal Name
		LED	J1	J2	JP4	J6	A (top)	B (bottom)		
	VCCO_0						1	1		SHIELD
	VCCO_0						2	2	GND	GND
TMS_B							3	3		TDO_XC2C
JTSEL							4	4		TCK_B
TDO_FX2							5	5	GND	GND
FX2_IO1	B4		◆			◆	6	6	GND	GND
FX2_IO2	A4		◆			◆	7	7	GND	GND
FX2_IO3	D5		◆			◆	8	8	GND	GND
FX2_IO4	C5		◆			◆	9	9	GND	GND
FX2_IO5	A6			◆		◆	10	10	GND	GND
FX2_IO6	B6			◆		◆	11	11	GND	GND
FX2_IO7	E7			◆		◆	12	12	GND	GND
FX2_IO8	F7			◆		◆	13	13	GND	GND
FX2_IO9	D7				◆	◆	14	14	GND	GND
FX2_IO10	C7				◆	◆	15	15	GND	GND
FX2_IO11	F8				◆	◆	16	16	GND	GND
FX2_IO12	E8				◆	◆	17	17	GND	GND
FX2_IO13	F9	LD7				◆	18	18	GND	GND
FX2_IO14	E9	LD6				◆	19	19	GND	GND
FX2_IO15	D11	LD5				◆	20	20	GND	GND
FX2_IO16	C11	LD4				◆	21	21	GND	GND
FX2_IO17	F11	LD3				◆	22	22	GND	GND
FX2_IO18	E11	LD2				◆	23	23	GND	GND
FX2_IO19	E12	LD1					24	24	GND	GND
FX2_IO20	F12	LD0					25	25	GND	GND
FX2_IO21	A13						26	26	GND	GND
FX2_IO22	B13						27	27	GND	GND
FX2_IO23	A14						28	28	GND	GND
FX2_IO24	B14						29	29	GND	GND
FX2_IO25	C14						30	30	GND	GND
FX2_IO26	D14						31	31	GND	GND
FX2_IO27	A16						32	32	GND	GND
FX2_IO28	B16						33	33	GND	GND
FX2_IO29	E13						34	34	GND	GND

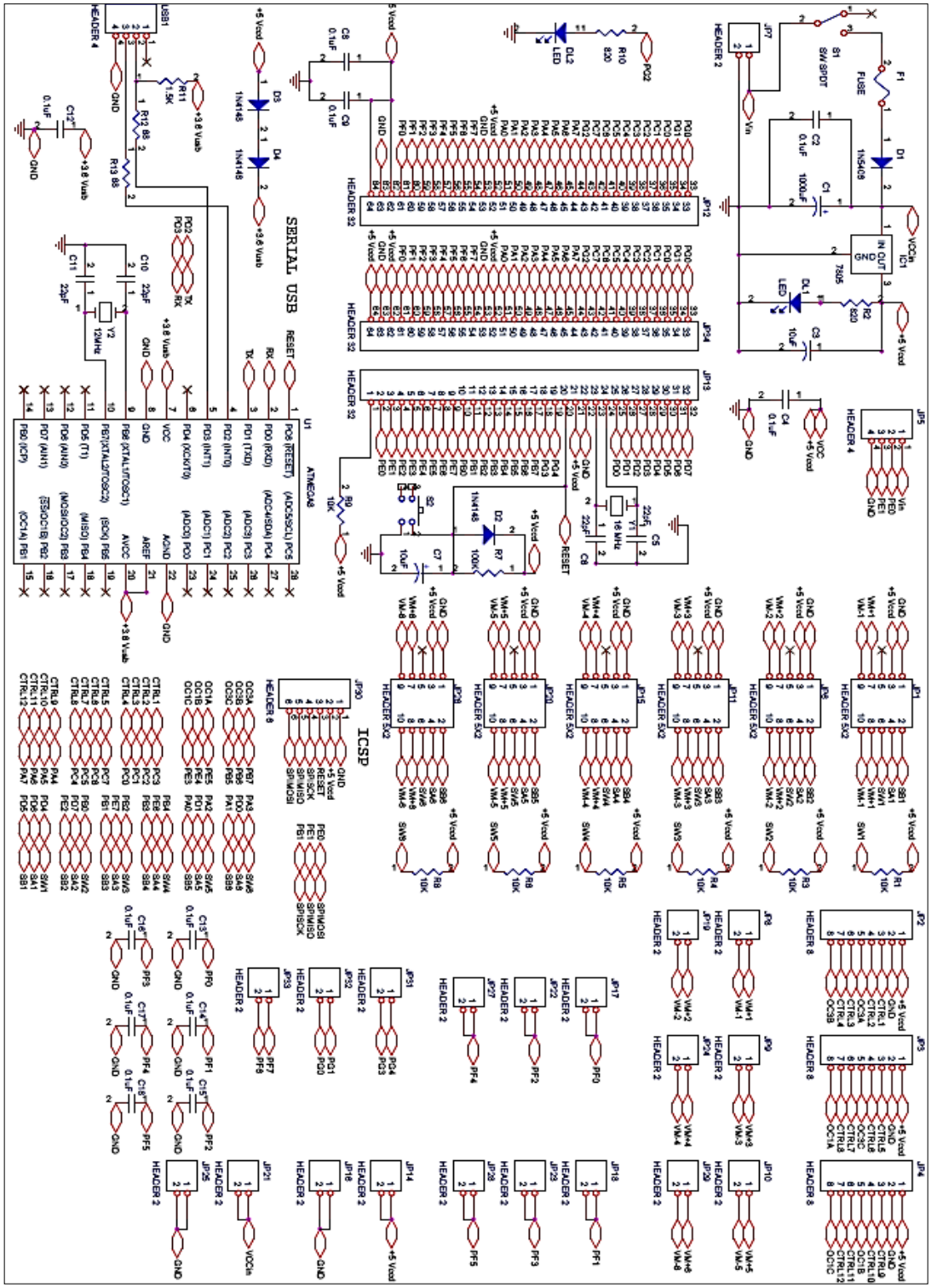
Anexo 2. Continuación distribución de pines tarjeta Spartan 3E

FX2_IO30	C4						35	35	GND	GND
FX2_IO31	B11						36	36	GND	GND
FX2_IO32	A11						37	37	GND	GND
FX2_IO33	A8						38	38	GND	GND
FX2_IO34	G9						39	39	GND	GND
FX2_IP35	D12						40	40	GND	GND
FX2_IP36	C12						41	41	GND	GND
FX2_IP37	A15						42	42	GND	GND
FX2_IP38	B15						43	43	GND	GND
FX2_IO39	C3						44	44	GND	GND
FX2_IP40	C15						45	45	GND	GND
GND	GND						46	46	E10	FX2_CLKIN
FX2_CLKOUT	D10						47	47	GND	GND
GND	GND						48	48	D9	FX2_CLKIO
5.0V							49	49		5.0V
5.0V							50	50		SHIELD

Anexo 3. Conexiones de pines del atmega 128



Anexo 4. Circuito del controlador



Anexo 5. Datasheet del motor pittman serie 8000

SERIES GM8000

Gearmotor Data

Line No.	Parameter	Symbol	Units	Reduction Ratios 6.3:1	9.9:1	19.5:1	30.9:1	60.5:1	95.9:1	187.7:1	297.5:1	581.8:1	922.3:1	1803.6:1
MECHANICAL SPECIFICATIONS														
1	Max. Load Standard Gears[1]	T_L	oz-in (N-m)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)	100 (0.71)
2	Max. Load Cut Steel Gears[1]	T_L	oz-in (N-m)	N/A (N/A)	160 (1.13)	160 (1.13)	160 (1.13)	160 (1.13)	160 (1.13)	160 (1.13)	160 (1.13)	160 (1.13)	160 (1.13)	160 (1.13)
3	Max. Load Wide Face Gears[1]	T_L	oz-in (N-m)	N/A (N/A)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)	175 (1.24)
4	Gearbox Shaft Rotation[2]	—	—	CW	CCW	CCW	CW	CW	CCW	CCW	CW	CW	CCW	CCW
5	Gearbox Efficiency	—	%	81	73	73	66	66	59	59	53	53	48	48
6	Gearbox Weight (Mass)	W_G	oz (g)	2.35 (66.6)	2.49 (70.6)	2.49 (70.6)	2.62 (74.3)	2.62 (74.3)	2.76 (78.2)	2.76 (78.2)	3.11 (88.2)	3.11 (88.2)	3.25 (92.1)	3.25 (92.1)
7	Gearbox Length	L_2	in max (mm max)	0.968 (24.6)	0.968 (24.6)	0.968 (24.6)	0.968 (24.6)	0.968 (24.6)	0.968 (24.6)	0.968 (24.6)	1.164 (29.6)	1.164 (29.6)	1.164 (29.6)	1.164 (29.6)
8	Length, GM82X2	L_3	in max (mm max)	2.977 (75.6)	2.977 (75.6)	2.977 (75.6)	2.977 (75.6)	2.977 (75.6)	2.977 (75.6)	2.977 (75.6)	3.173 (80.6)	3.173 (80.6)	3.173 (80.6)	3.173 (80.6)
9	Length, GM82X3	L_3	in max (mm max)	3.102 (78.8)	3.102 (78.8)	3.102 (78.8)	3.102 (78.8)	3.102 (78.8)	3.102 (78.8)	3.102 (78.8)	3.298 (83.8)	3.298 (83.8)	3.298 (83.8)	3.298 (83.8)
10	Length, GM82X4	L_3	in max (mm max)	3.352 (85.1)	3.352 (85.1)	3.352 (85.1)	3.352 (85.1)	3.352 (85.1)	3.352 (85.1)	3.352 (85.1)	3.548 (90.1)	3.548 (90.1)	3.548 (90.1)	3.548 (90.1)
11	Length, GM87X2	L_3	in max (mm max)	2.91 (73.9)	2.91 (73.9)	2.91 (73.9)	2.91 (73.9)	2.91 (73.9)	2.91 (73.9)	2.91 (73.9)	3.106 (78.9)	3.106 (78.9)	3.106 (78.9)	3.106 (78.9)
12	Length, GM87X3	L_3	in max (mm max)	3.035 (77.1)	3.035 (77.1)	3.035 (77.1)	3.035 (77.1)	3.035 (77.1)	3.035 (77.1)	3.035 (77.1)	3.231 (82.1)	3.231 (82.1)	3.231 (82.1)	3.231 (82.1)
13	Length, GM87X4	L_3	in max (mm max)	3.285 (83.4)	3.285 (83.4)	3.285 (83.4)	3.285 (83.4)	3.285 (83.4)	3.285 (83.4)	3.285 (83.4)	3.481 (88.4)	3.481 (88.4)	3.481 (88.4)	3.481 (88.4)
NO-LOAD SPEED														
14	GM8X22	S_{NL}	rpm (rad/s)	1246 (130)	786 (82.3)	402 (42.1)	253 (26.5)	130 (13.6)	81.8 (8.57)	41.8 (4.38)	26.4 (2.76)	13.5 (1.41)	8.51 (.891)	4.35 (.456)
15	GM8X23	S_{NL}	rpm (rad/s)	1317 (138)	831 (87.0)	425 (44.5)	268 (28.1)	137 (14.3)	86.5 (9.06)	44.2 (4.63)	27.9 (2.92)	14.3 (1.50)	9.00 (.942)	4.60 (.482)
16	GM8X24	S_{NL}	rpm (rad/s)	1612 (169)	1017 (107)	520 (54.5)	328 (34.3)	168 (17.6)	106 (11.1)	54.1 (5.67)	34.1 (3.57)	17.5 (1.83)	11.0 (1.15)	5.63 (.590)

[1]Represents gearbox capability only. Continuous load torque capability will vary with gear ratio, motor selection, and operating conditions.
[2]Shaft rotation is designated while looking at output shaft with motor operating in a clockwise direction. Gearmotor is polarity reversible.

Motor Data

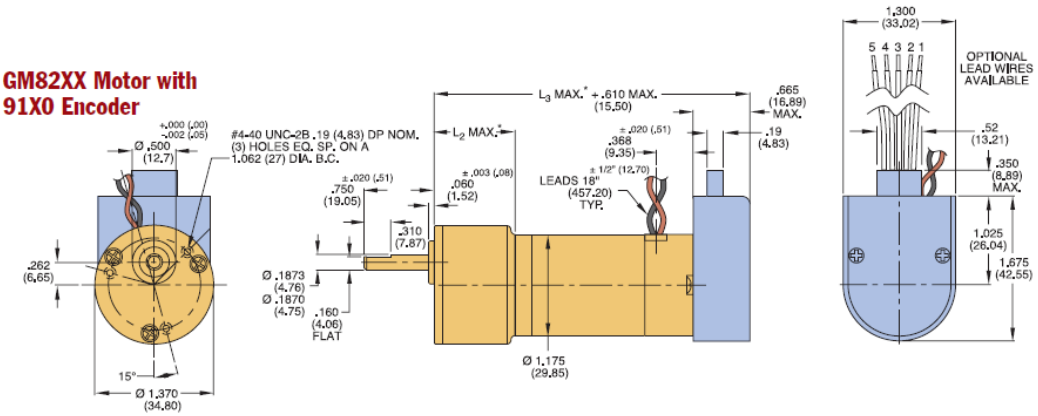
Line No.	Parameter	Symbol	Units	8X22	8X23	8X24
17	Continuous Torque (Max.)[3]	T_C	oz-in (N-m)	1.6 (11.2×10^{-3})	2.0 (14.1×10^{-3})	2.6 (18.5×10^{-3})
18	Peak Torque (Stall)	T_{PK}	oz-in (N-m)	7.4 (52.0×10^{-3})	10.5 (74.2×10^{-3})	16.8 (118.6×10^{-3})
19	Motor Constant	K_M	oz-in/√W (N-m/√W)	1.12 (7.9×10^{-3})	1.30 (9.2×10^{-3})	1.49 (710.5×10^{-3})
20	No-Load Speed	S_0	rpm (rad/s)	7847 (822)	8298 (869)	10158 (1064)
21	Friction Torque	T_F	oz-in (N-m)	0.35 (2.5×10^{-3})	0.35 (2.5×10^{-3})	0.35 (2.5×10^{-3})
22	Rotor Inertia	J_M	oz-in-s^2 (kg-m^2)	1.4×10^{-4} (9.89×10^{-7})	1.7×10^{-4} (1.20×10^{-6})	2.3×10^{-4} (1.62×10^{-6})

[3]Continuous torque specified at 25°C ambient temperature and without additional heat sink.

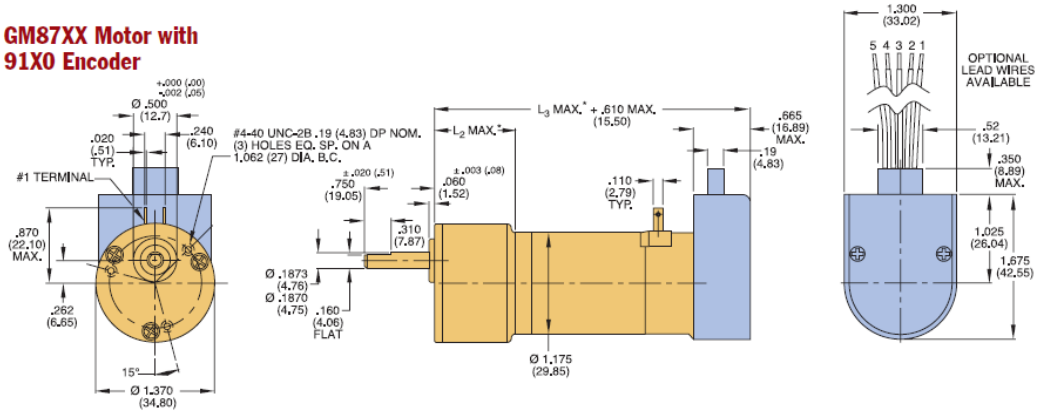
PITTMAN®

LCG-2

GM82XX Motor with 91X0 Encoder



GM87XX Motor with 91X0 Encoder



Encoder Connection Chart

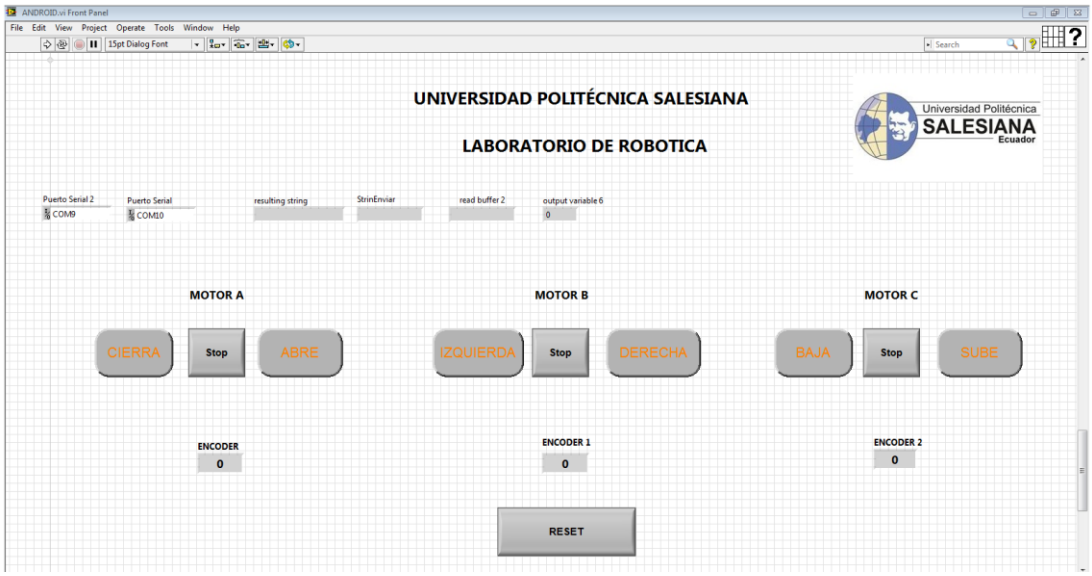
Pin No.	Color	Connection
1	Black	Ground
2	Green	Index/NC
3	Yellow	Channel A
4	Red	Vcc
5	Blue	Channel B

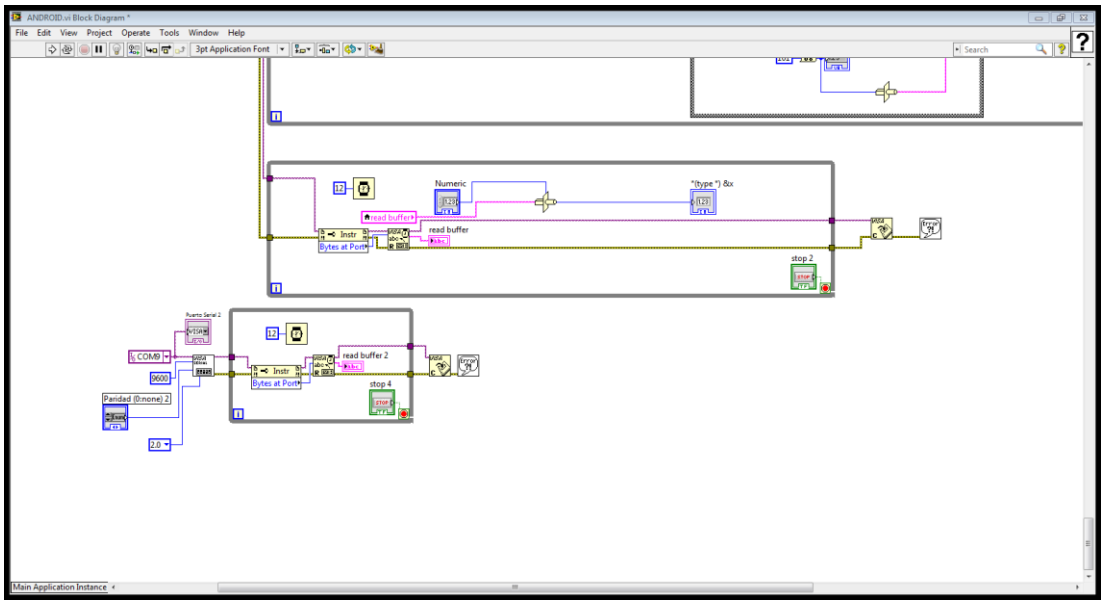
Notes:
- Unless otherwise specified, all tolerances are to be ±.005 (.01)
- All measurements are in inches (mm)

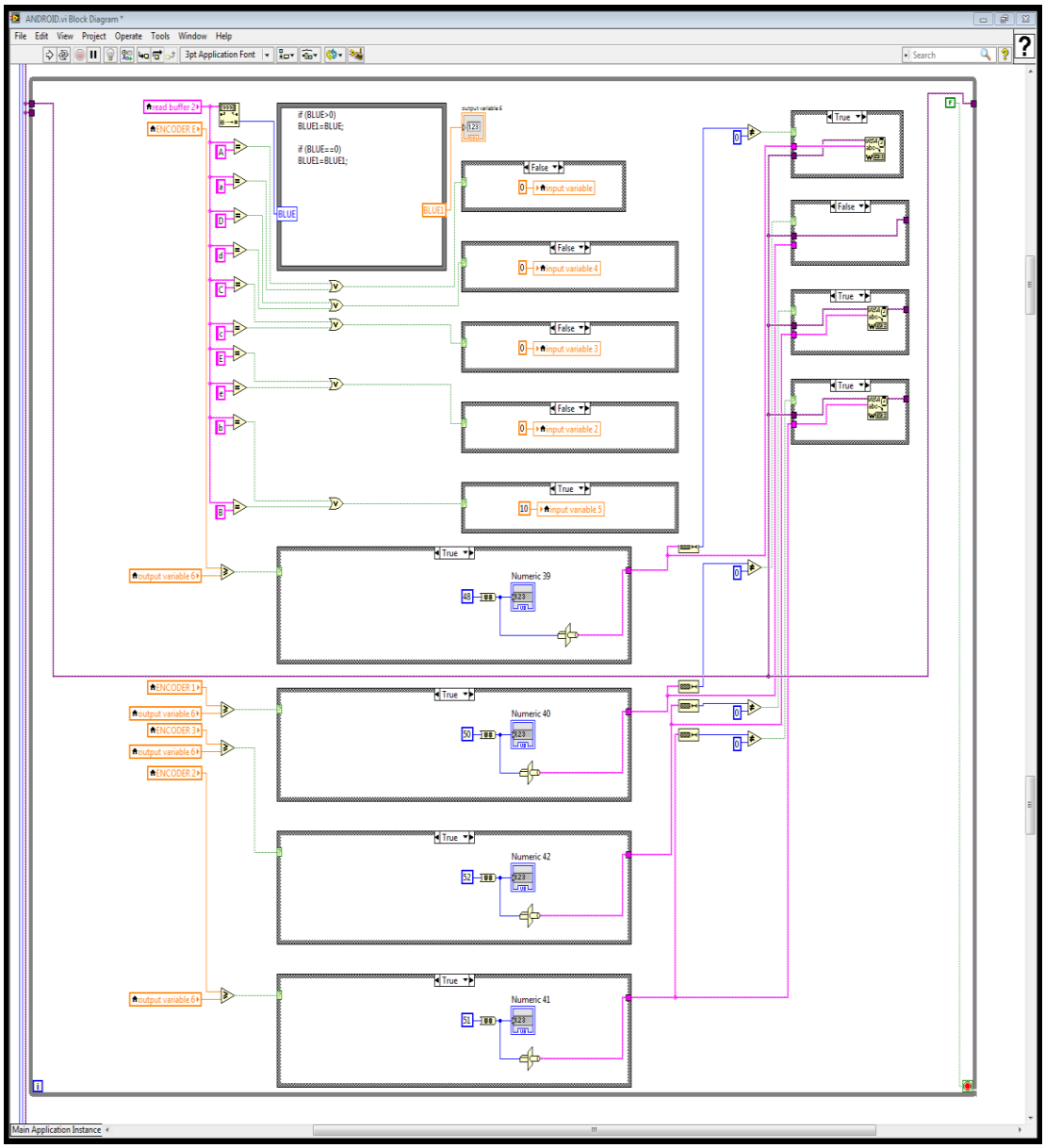
*See line numbers 7 through 13 in gearmotor data chart

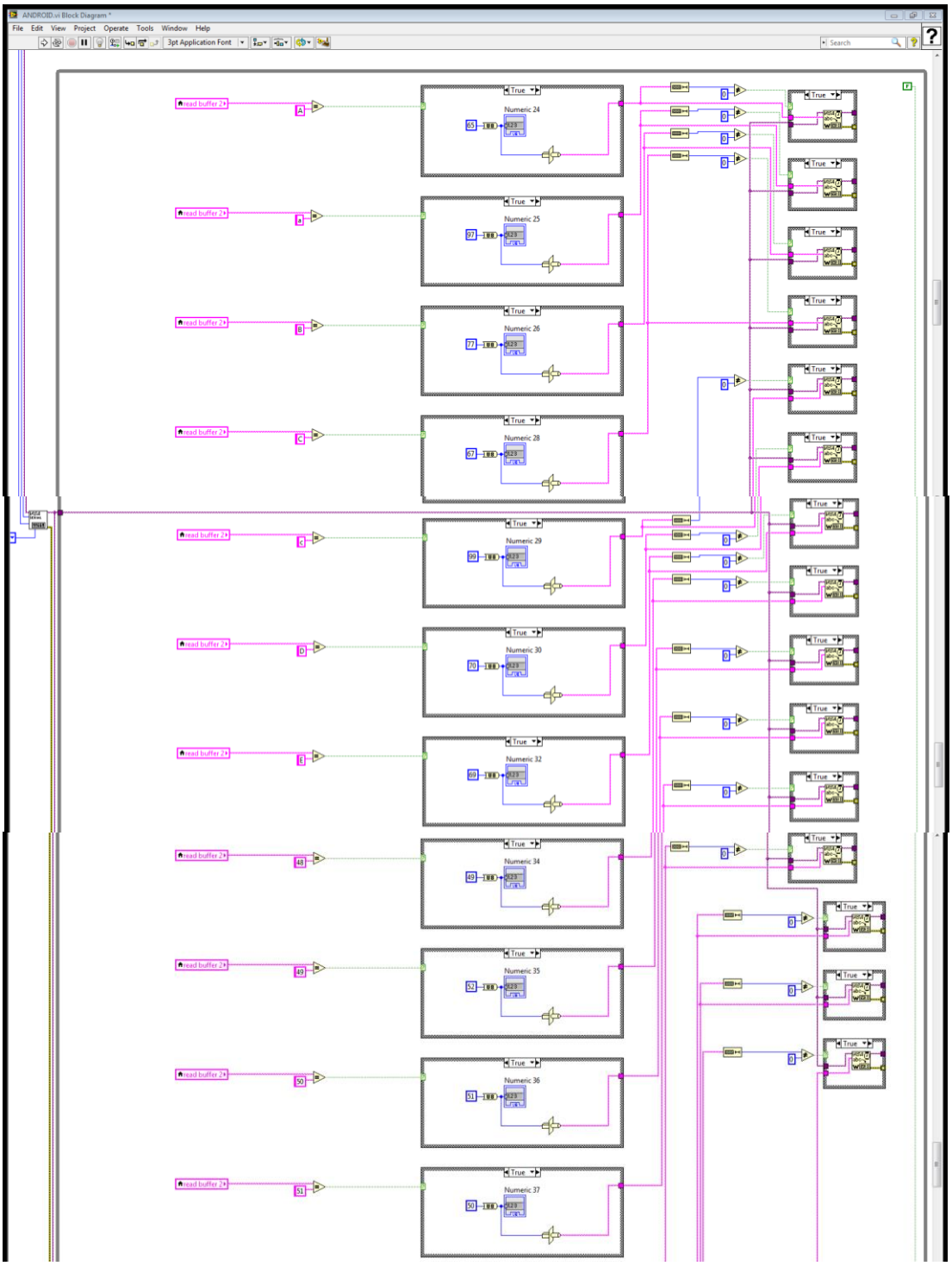
PITTMAN®

Anexo 6. Control del brazo robótico en LabVIEW vía bluetooth

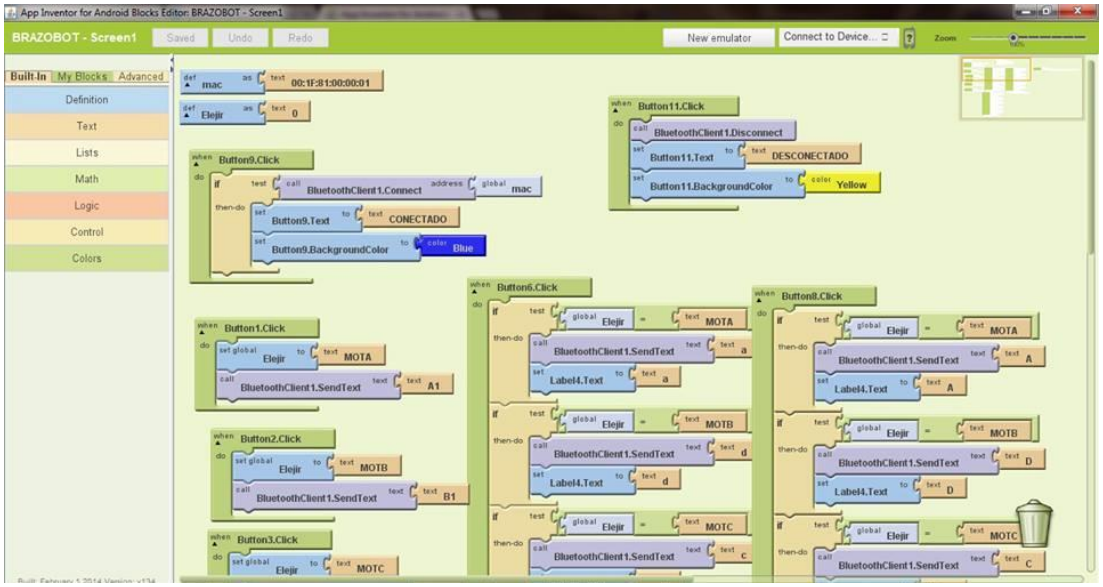


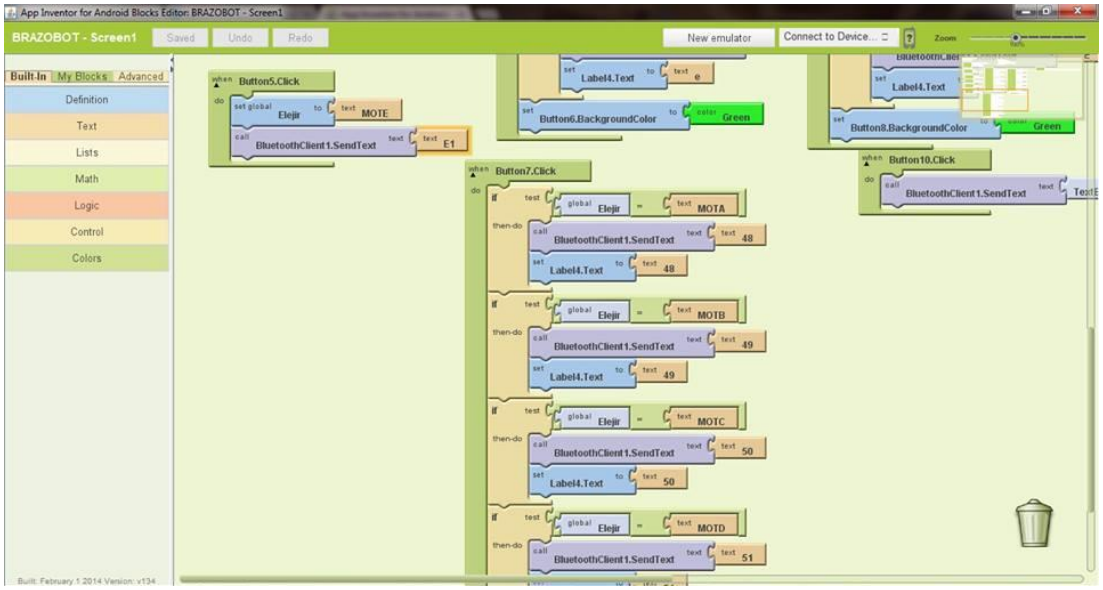


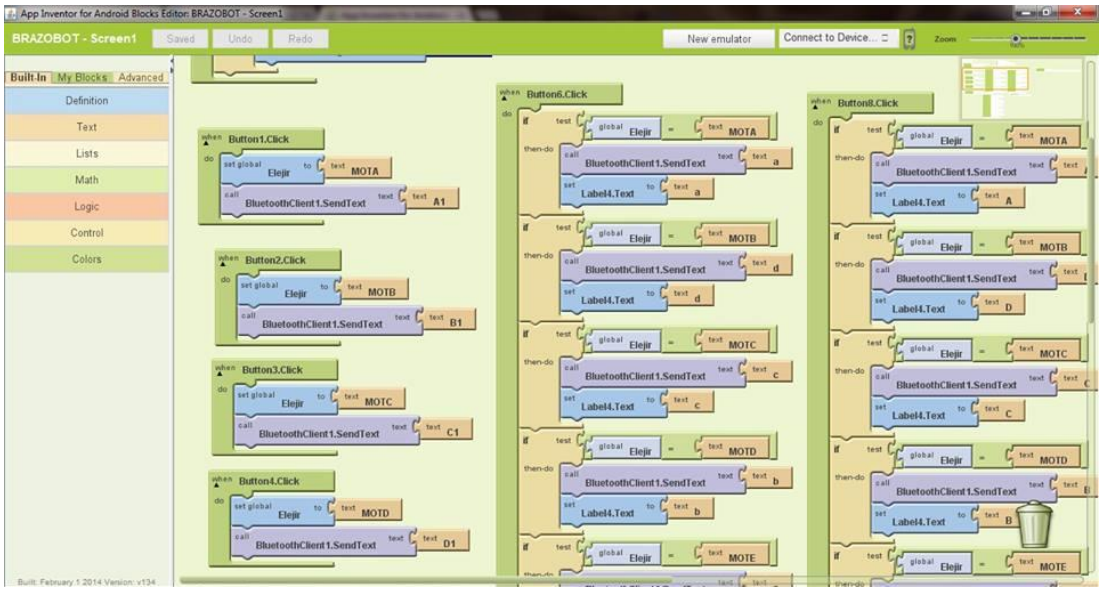

ANDROID.vi Block Diagram *
File Edit View Project Operate Tools Window Help
3pt Application Font
Search
read buffer 2
ENCODER 0
if (BLUE>0)
BLUE1=BLUE;
if (BLUE==0)
BLUE1=BLUE;
BLUE
BLUE1
output variable 6
False
input variable
input variable 4
input variable 3
input variable 2
True
input variable 5
Numeric 39
Numeric 40
Numeric 42
Numeric 41
ENCODER 1
ENCODER 3
ENCODER 2
Main Application Instance

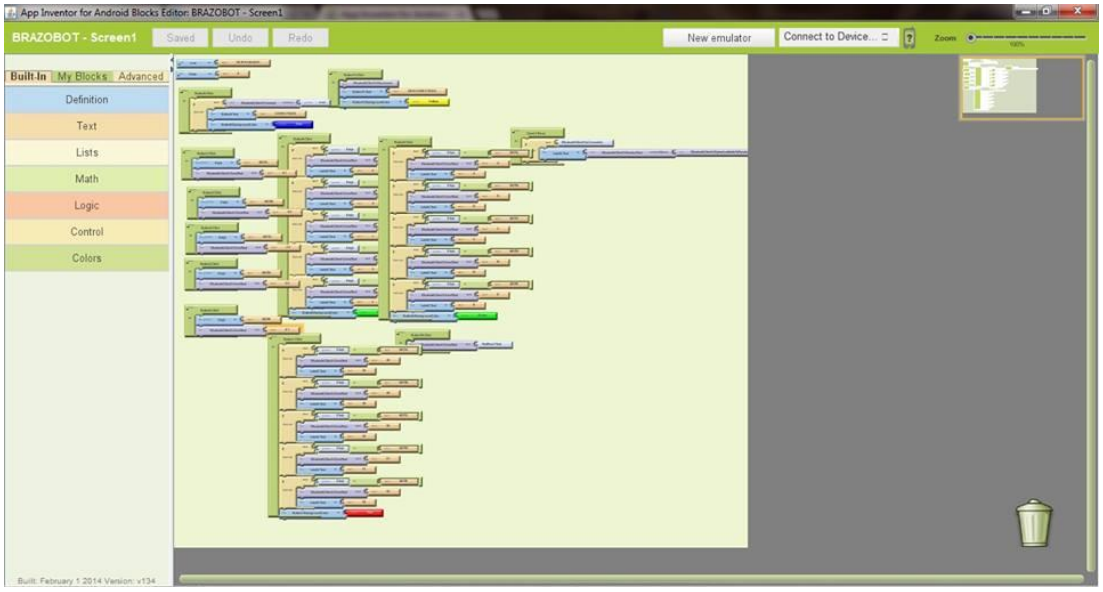

ANDROID.vi Block Diagram *
File Edit View Project Operate Tools Window Help
3pt Application Font
Search
read buffer 2
A
B
C
D
E
48
49
50
51
Numeric 24
Numeric 25
Numeric 26
Numeric 28
Numeric 29
Numeric 30
Numeric 32
Numeric 34
Numeric 35
Numeric 36
Numeric 37
True

Anexo 7. Creación de la aplicación Android para control del brazo robótico






App Inventor for Android Blocks Editor: BRAZOBOT - Screen1
BRAZOBOT - Screen1
Saved
Undo
Redo
New emulator
Connect to Device...
Zoom
Built-In
My Blocks
Advanced
Definition
Text
Lists
Math
Logic
Control
Colors
Button1.Click
Button2.Click
Button3.Click
Button4.Click
Button6.Click
Button8.Click
Elejir
MOTA
MOTB
MOTC
MOTD
MOTE
A1
B1
C1
D1
BluetoothClient1.SendText
Label4.Text


App Inventor for Android Blocks Editor: BRAZOBOT - Screen1
BRAZOBOT - Screen1
Saved
Undo
Redo
New emulator
Connect to Device...
Zoom
Built-In
My Blocks
Advanced
Definition
Text
Lists
Math
Logic
Control
Colors