ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"BRAZO ROBÓTICO DIDÁCTICO USANDO PLATAFORMA NIOS II"

TESINA DE SEMINARIO

Previa la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

INGENIERO EN TELEMÁTICA

INGENIERO EN TELEMÁTICA

Presentado por:

Eder Antonio Vargas Vargas

Iván Alejandro Sandoya Tinoco

Enrique Villalta Fernández

GUAYAQUIL – ECUADOR

AÑO 2014

AGRADECIMIENTO

Quiero agradecer a todas y cada una de las personas que ayudaron a poder terminar con uno de los objetivos que me he planteado, en especial a Dios que es la razón de mi existencia, a mis padres que han sido testigo de mi esfuerzo pese a las dificultades y a mi hermano que ha sido mi fuerza para siempre mirar hacia adelante.

También quisiera dar la gracias a todos mis compañeros que tuve la satisfacción de conocer durante el trayecto de mi formación universitario, ya que con su ayuda me han ayudado a crecer como ser humano y en conocimientos.

Iván Alejandro Sandoya Tinoco.

En primer lugar agradezco a Dios, porque sin él nada fuera posible.

Agradezco a mis padres, quienes día a día me motivaron a seguir adelante, y además me enseñaron que todo sacrificio conlleva a una recompensa.

Agradezco a mis amigos y demás familiares con los cuales pude aprender a ser una mejor persona y de buenos valores.

Enrique Villalta Fernández.

Agradezco a Dios quien me ha dado la oportunidad de ser una persona de bien, de crecer en mente y espíritu, permitiendo alcanzar este logro.

A mis Padres quienes han sido el pilar fundamental guiándome en el transcurso de mi vida académica, especialmente a mi Madre quien no dejo flaquear para obtener este título profesional, ahora entiendo estas palabras: “La Alegría De Un Padre Es Ver A Sus Hijos Profesionales”.

A mis hermanos, familiares y amigos que de alguna u otra manera forman parte de la Escuela de la Vida, quienes han estado complementando mi formación.

Eder Antonio Vargas Vargas.

DEDICATORIAS

A mis padres que me han ayudado en todo mi vida académica para poder cumplir mi objetivo que ha sido terminar mi formación profesional.

Iván Alejandro Sandoya Tinoco.

A mis padres por toda su ayuda brindada ya que sin ellos nada de lo obtenido podría haber sido posible.

Enrique Villalta Fernández.

Dedico esta tesina a mis padres Milton Vargas y Fanny Vargas, quienes han estado apoyando a en todo momento con sus consejos que han sido fructíferos para alcanzar esta meta de culminar mis estudios Superiores.

También le dedico a mi hermano Joel quien aunque a la distancia siempre está pendiente de mi apoyándome incondicionalmente y mi hermano Cristian con quien he convivido durante esta gran cruzada.

Eder Antonio Vargas Vargas.

TRIBUNAL DE SUSTENTACIÓN

Ing. Ronald Ponguillo

PROFESOR DEL SEMINARIO DE GRADUACIÓN

Ing. Víctor Asanza

PROFESOR DELEGADO POR EL DECANO DE LA FACULTAD

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este trabajo, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL". (Reglamento de exámenes y títulos profesionales de la ESPOL)

_____

Iván Alejandro Sandoya Tinoco.

_____

Enrique Villalta Fernández.

_____

Eder Antonio Vargas Vargas.

RESUMEN

Este proyecto consiste en el diseño e implementación de un algoritmo usando la tarjeta de desarrollo **DE0-Nano** de Cyclone IV de Altera**,** que nos va a permitir controlar los movimientos del brazo robótico de forma inalámbrica en modo manual o automático.

Para el desarrollo del proyecto fue necesario agregar a la mini computadora diseñada a través del software Qsys el componente de modulación por ancho de pulso denominado PWM, que nos permite controlar las posiciones angulares del conjunto de servos motores que utiliza nuestro brazo robótico.

Además, el uso de los puertos de expansión de la tarjeta DE0-Nano de Altera, para la comunicación entre el micro-procesador del sistema y el módulo emisor-receptor ultrasónico, así como también el manejo de la interfaz de la pantalla LCD 2x16 (filas x columnas). Al proyecto se lo ha estructurado en 4 capítulos como se lo explica a continuación:

En el **primer capítulo**, se exponen los objetivos generales y específicos, así como también los alcances y limitaciones del proyecto.

En el **segundo capítulo**, damos a conocer a fondo cada uno de los elementos que se han usado, explicando los conceptos básicos para el desarrollo e implementación del sistema, así como el uso de la herramienta de desarrollo que ofrece Google para la creación de aplicaciones basada en Android.

En el **tercer capítulo**, se presenta la implementación del proyecto y el diseño de la interconexión de los elementos para poner en a funcionar el sistema en conjunto.

En **cuarto capítulo**, se presentan los resultados de las pruebas realizadas, demostrando así el buen funcionamiento de brazo robótico.

ÍNDICE GENERAL

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 3

CAPÍTULO 4

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS

ADC	Convertidor Analógico-Digital
ASIC	Circuito Integrado de Aplicación Específica
CMOS	Semiconductor de Óxido Metálico Complementario
E/S	Entrada y Salida
FPGA	Campo Matriz de Puertas Programables
GPIO	Entradas/salidas de Propósito General
HDL	Lenguaje de Descripción de Hardware
ISM	Industria Científica y Médica
JTAG	Grupo de Acción Conjunta de Prueba
LCD	Pantalla de Cristal Líquido
LED	Diodo Emisor de Luz
PCB	Placa de Circuito Impreso
PWM	Modulación por Ancho de Pulso

RS-232	Recommended Standard 232
SBT	Herramientas de Construcción Software para Eclipse
SDRAM	Memoria de Acceso Aleatoria Dinámica Sincrónica
SOPC	Sistema en Chip Programable
UART	Transmisor – Receptor Universal Asincrónico
UHF	Frecuencia Ultra Alta
USB	Bus Serial Universal
VHDL	Lenguaje de Descripción de Hardware de Alto nivel

INTRODUCCIÓN

Hoy en día debido al gran avance tecnológico se ha logrado diseñar circuitos digitales capaces de realizar procesos a gran velocidad, además de poseer una alta capacidad de almacenamiento. Esto ha cobrado mucha importancia entre los usuarios ya que nos permite desarrollar proyectos basados en los dispositivos FPGA (Field Progarmmable Gate Array). Estos dispositivos configurables permiten al usuario programar y diseñar de acuerdo a las necesidades y requerimientos de su sistema.

Nuestro proyecto consiste en el diseño e implementación de un brazo robótico con 6 grados de libertad, capaz de recolectar objetos siguiendo un patrón establecido de manera autónoma y manual. Para la interacción con el usuario se ha desarrollado una interfaz intuitiva que trabaja sobre el sistema operativo Android, que actualmente se encuentra en la mayoría de dispositivos inteligentes (Tablets y Smartphones).

Como herramienta de desarrollo de nuestra aplicación se usó MIT App Inventor en su versión beta, destacada hoy en día por ser la primera en su tipo que usa el lenguaje G, lo cual otorga al desarrollador una configuración transparente con el dispositivo usando abstracción de nivel superior.

CAPÍTULO 1

1. GENERALIDADES

En este capítulo se da a conocer cuáles son los objetivos generales y específicos, así como también los alcances y limitaciones del proyecto.

1.1. OBJETIVOS

1.1.1. OBJETIVOS GENERALES

Implementar un algoritmo estratégico en NIOS II SBT que pueda controlar el movimiento de un brazo robótico, capaz de detectar y recoger un determinado tipo de objeto que se encuentre dentro de su campo de acción.

1.1.2. OBJETIVOS ESPECÍFICOS

- Aprender el uso y manejo de la Tarjeta DE0-Nano de Altera.
- Desarrollar un sistema embebido basado en FPGA, el cual posea los componentes necesarios para poder llevar a cabo el control del brazo robótico.
- Aprender el uso de algoritmos para la detección de objetos usando un sensor ultrasónico HC-SR04.
- Aprender el uso de algoritmos para el control de una pantalla LCD 16X2.

- Establecer un enlace de comunicación vía Bluetooth, que permita la transferencia de información entre el sistema embebido y la interfaz usuario de forma remota.

- Elaborar un software en el entorno de desarrollo de NIOS II SBT, capaz de administrar eficientemente los recursos existentes del sistema embebido.

- Desarrollar los circuitos necesarios para poder comunicar nuestra tarjeta DE0-Nano con el brazo robótico.

- Crear una Aplicación que trabaje sobre la plataforma Android, que permita controlar de forma inalámbrica el Brazo Robótico.

1.2. IDENTIFICACIÓN DEL PROBLEMA

Según estudios realizados se sabe que el ser humano sólo puede trabajar a todo su potencial por determinado número de horas, después de ese período se necesita un tiempo de descanso para poder retomar las labores, ya que empieza a perder concentración por el cansancio. Nuestro proyecto busca suplir esta desventaja, con un robot capaz trabajar por períodos de tiempo prolongados y cumplir objetivos de una forma más precisa.

Además de lo mencionado se ha planteado eliminar la manipulación de sustancias u objetos perjudiciales para la salud. Como por ejemplo, el personal que clasifica la basura que siempre está expuesto a contraer cualquier tipo de enfermedad, por lo que con nuestro proyecto del brazo robótico se podría reemplazar por las manos del ser humano para evitar alguna enfermedad.

1.3. METODOLOGÍA

La metodología a usar para el cumplimiento de cada uno de nuestros objetivos, es diseñar un brazo robótico de 6 grados de libertad, para la parte de control, los criterios de diseño se enfocaron en la facilidad de adquisición, y versatilidad para acoplarse a los requerimientos de nuestro proyecto.

Los grados de libertad son representados por servomotores que tiene la capacidad de ubicarse en un rango de operación y mantenerse estable en su posición, éstos usan la modulación por ancho de pulsos (más conocido como PWM en inglés) para controlar dirección y posición. Su electrónica interna responde al ancho de la señal modulada para moverse dentro de su campo de acción, para lo cual se crea un componente diseñado en NIOS II capaz de modular una señal por los puertos de entrada y salida de nuestra tarjeta DE0-Nano.

Para el acoplamiento de la tarjeta de desarrollo y el brazo robótico, específicamente con los servomotores, se diseñó una etapa de fuerza, para proteger la DE0-Nano de cambios bruscos de corrientes o voltajes que ocurra en el funcionamiento del sistema en conjunto.

Para la interfaz de usuario se ha usado una aplicación para dispositivos móviles con sistema Android, que permite enviar instrucciones al brazo robótico de forma inalámbrica mediante Bluetooth.

Además se utiliza una pantalla LCD, la cual es una venta de comunicación del prototipo hacia el usuario mostrando información sobre el funcionamiento como por ejemplo el modo en que se encuentra trabajando.

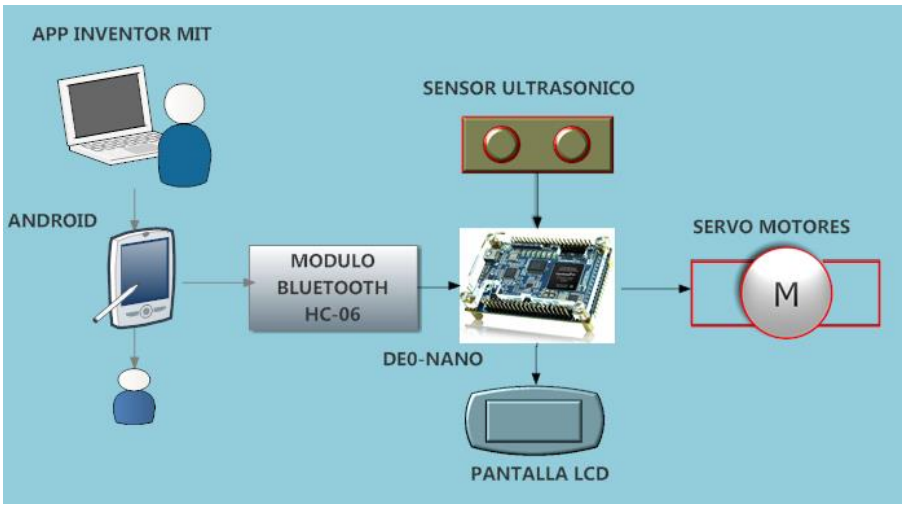


Figura 1.1 Diagrama de Bloques del Proyecto

1.4. ALCANCES Y LIMITACIONES DEL PROYECTO

Entre los alcances del proyecto se tiene:

- ✓ El brazo robótico en modo automático, es capaz de detectar la presencia de tres objetos que se encuentran en posiciones predeterminadas y colocarlas en un lugar establecido.
- ✓ El sistema de detección verifica la existencia de un objeto por la variación de altura entre el campo de acción del brazo robótico y el sensor ultrasónico.
- ✓ El brazo robótico es ubicado en un punto estratégico y fijo de tal forma que el campo de acción sea el óptimo.
- ✓ El movimiento del brazo incorpora pequeños retardos de tiempo, con la intención de que los desplazamientos no sean bruscos y puedan dañar los engranajes de los servomotores.
- ✓ Todo dispositivo portable que tenga instalado el sistema operativo Android y además posea la tecnología Bluetooth, es capaz de controlar el brazo robótico. Permitiendo al usuario elegir con que dispositivo móvil controlar el prototipo.

Entre las limitaciones se tienen las siguientes:

- ✓ El campo de acción está limitado por el tamaño del brazo robótico.
- ✓ Los servomotores que sirven como articulación están restringidos en ángulos de 0º a 180º.
- ✓ El tamaño del objeto se encuentra limitado a la abertura máxima que tiene pinza del brazo o a la tenaza; además debe tener una superficie rugosa para que no resbalen con facilidad una vez que la tenaza se cierre.
- ✓ El peso del objeto se encuentra restringido a la fuerza que imprimen los servomotores para mantener la estructura metálica posicionada, ya que un peso mayor a éste puede provocar fallos en el funcionamiento o el daño irreparable de cualquiera de los servomotores.

CAPÍTULO 2

2. MARCO TEÓRICO

Este capítulo damos a conocer de una forma más detallada cada uno de los elementos usados para interacción usuario-robot y características de la tarjeta DE0-Nano Cyclone IV de Altera, además del software que usamos para el desarrollo del proyecto.

2.1. SISTEMAS EMBEBIDOS CONFIGURABLES

Con el tiempo las tecnologías han ido evolucionando día a día desde un simple transistor hasta sistemas embebidos muy complejos, pero fácil de programar. Durante el estudio de la ingeniería electrónica fue necesario el diseño y montaje de circuitos digitales para entender su funcionamiento.

Luego surge la necesidad de utilizar un sistema que permita simular estos circuitos, esta característica se presenta en las FPGAs y CPLDs que son dispositivos lógicos programables. Los sistemas embebidos configurables están basados en FPGAs, que contiene bloques de lógica programable que facilita su interconexión y son configurados mediante el lenguaje descripción de hardware (por sus siglas en ingles HDL).

2.1.1. TARJETA DE DESARROLLO DE0-NANO DE ALTERA

La **DE0-Nano** de Altera fue diseñada con fines didácticos en lo que concierne al diseño y desarrollo digital de circuitos, ya que fue desarrollada por profesores que debido a sus experiencias vieron necesario el uso de una herramienta que ayude a sus alumnos crear pequeños proyectos reales de la vida diaria.

La tarjeta introduce una compacta **FPGA Cyclone IV** (con 22,320 elementos lógicos) que generalmente se utiliza para diseños de circuitos en prototipo como robots y proyectos “portables”. (AlteraAltera Corporation, 1995 - 2014)

La tarjeta trae consigo:

- La tarjeta DE0-Nano de 3x6’ con la FPGA Cyclone IV EP4C22 (256-pin).
- Cable USB.
- Cobertor Plexiglás para la tarjeta.
- Guía de instalación.

Características	Descripción
FPGA	Cyclone IV EP4CE22F17C6N with EPCS64 64-Mbits serial configuration device
Interfaces E/S	Cable USB-Blaster para la configuración de la FPGA
	Acelerómetros para los 3-ejes con 13-bits de resolución
	Convertidores Analógicos-Digitales, 8 canales, 12-bits de resolución
	Cabecera de Expansión (dos de 40-pin y una 26-pin)
	Cabecera Externas de Fuente (2-pin)
Memoria	32 MB SDRAM
	2 Kb EEPROM
Switches and LEDs	8 LEDs verdes
	4 dip switches
	2 botoneras
Clock	50 MHz clock

Tabla 2.1 Información sobre la Tarjeta DE0-Nano de Altera

Figura 2.1 Tarjeta DE0-Nano de ALTERA

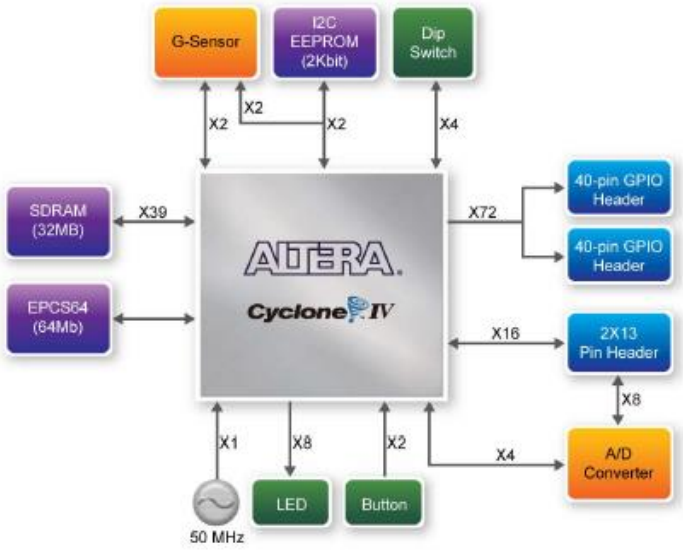


Figura 2.2 Diagrama de Bloques de la DE0-Nano

2.1.2. PROCESADOR NIOS II

NIOS II es una infraestructura de procesador de propósito general que puede ser configurado usando algún HDL para varias FPGA de Altera.

Las características principales de este procesador serán descritas a continuación:

- Tamaño de palabra de 32 bits.
- Juego de instrucciones RISC de 32 bits.
- 32 registros de propósito general de 32 bits (r0 – 31).
- 6 registros de control de 32 bits (ctl0 - ctl5).
- 32 fuentes de interrupción externa.
- Capacidad de direccionamiento de 32 bits.
- Operaciones de multiplicación y división de 32 bits.
- Instrucciones dedicadas para multiplicaciones de 64 y 128 bits.
- Instrucciones para operaciones de coma flotante en precisión simple.
- Acceso a variedad de periféricos integrados e interfaces para manejo de memorias y periféricos externos.

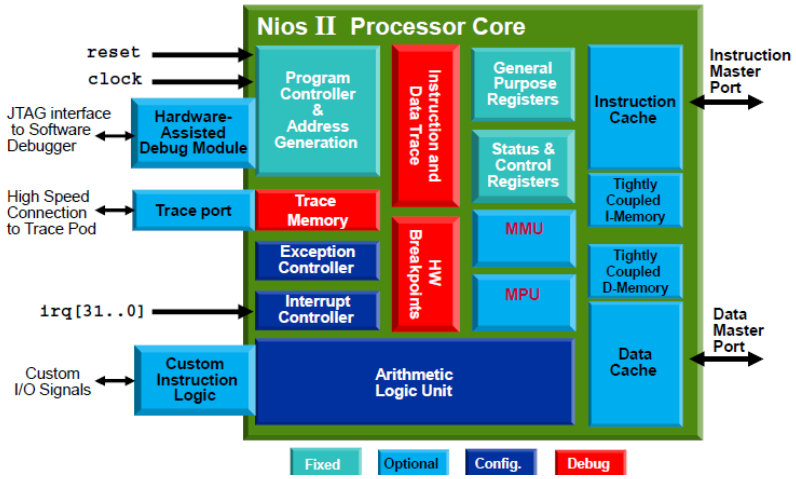


Figura 2.3 Diagrama de Bloques del microprocesador NIOS II

Dependiendo como se quiera maximizar el rendimiento del procesador u optimizar los recursos del FPGA existen tres versiones que son:

- **El NIOS II/f (fast)** es la versión diseñada para el alto rendimiento y que con un pipeline de 6 etapas proporciona opciones específicas para aumentar su desempeño, como memoria caché de instrucciones y datos o unidad de manejo de memoria.

- **El NIOS II/s (standard)** es la versión con pipeline de 5 etapas que dotada de una unidad aritmética lógica ALU busca combinar rendimiento y consumo de recursos.

- **El NIOS II/e (economy)** es la versión que requiere menos recursos de la FPGA, sin pipeline y muy limitada, dado que carece de las operaciones de multiplicación y división.

2.1.3. QUARTUS II

QUARTUS II es un software desarrollado por Altera para diseño completo con dispositivos lógicos programables CPLDs y FPGAs. Posee ciertas ventajas como:

- Un editor de lenguaje simbólico.
- Un compilador que puede como entrada recibir archivos VHDL, VERILOG, también archivos con circuitos esquemáticos. Puede ejecutar simulaciones, usando como archivo de entrada uno creado por el editor de formas de onda.

- Puede programar un dispositivo FPGA especificando previamente su modelo.

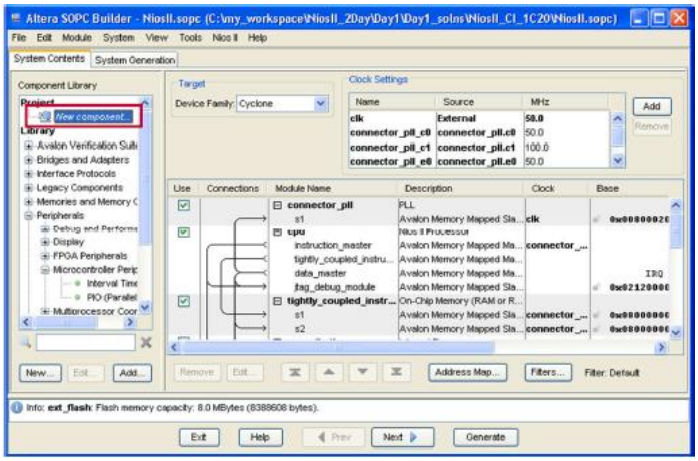


Figura 2.4 Interfaz SOPC Builder

Quartus II esta enlazado a una aplicación llamada SOPC Builder (Figura 2.4), es una herramienta propia de Altera la cual realiza la conexión interna de componentes de hardware. En SOPC Builder viene incorporado con una biblioteca de componentes o IP CORES predefinidos, tales como controladores memoria, periféricos y el microprocesador NIOS II.

2.1.4. NIOS II SOFTWARE BUILD TOOLS FOR ECLIPSE

Es una herramienta de desarrollo de software para la familia de microprocesadores embebidos NIOS II. Se programa en lenguaje C y éste genera un archivo SOPC Builder que se graba en la mini-computadora básica de la tarjeta DE0-Nano.

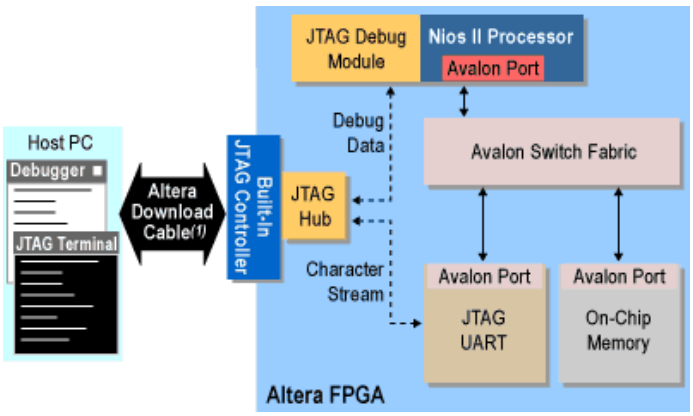


Figura 2.5 Conexión de NIOS II con la DE0-Nano

2.2. ROBÓTICA

2.2.1. DEFINICIÓN

La robótica es una ciencia que se encarga del diseño, la fabricación y la utilización de robots para desempeñar tareas comúnmente realizadas por el ser humano, obteniendo resultados más eficaces. Un robot es considerado como una máquina que puede ser programada con las nuevas tecnologías de tal forma que interactúe usando objetos e imite el comportamiento del ser humano.

En la robótica se combinan muchas ciencias tales como ingeniería en electrónica, informática y mecánica que son ejes fundamentales de la robótica. (Definicion.de, 2008-2014)

Figura 2.6 Rossum's Universal Robots

La Figura 2.6 muestra un robot de una obra checoslovaca publicada en 1917 por Karel Kapek, llamada Rossum’s Universal Robots, que dio lugar al nombre de robot, viene la palabra checa “Robota” que significa servidumbre o trabajador forzado, al traducirlo al inglés se convirtió en el término robot. (Jose Agraszal, 2010)

2.2.2. EL BRAZO ROBÓTICO

En la actualidad la robótica ha evolucionado de forma acelerada, tanto así que ha dado lugar al desarrollo de nuevas disciplinas como la cirugía robótica, que tiene

como objetivo principal mejorar la salud del ser humano. Con el uso apropiado de robots sofisticados y de precisión se logra obtener un porcentaje de error mínimo. Es por esto que se ha dado como una alternativa muy fiable la utilización de un brazo robótico en algunos procesos quirúrgicos.

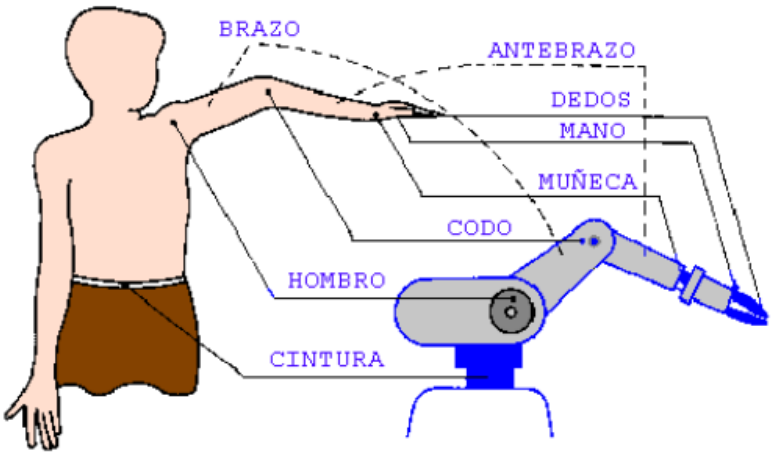


Figura 2.7 Semejanza de un Brazo Robótico con la Anatomía Humana

2.3. SERVOMOTORES

Un servomotor es similar a un motor de corriente continua también conocido como servo, que tiene la capacidad de fijarse en una posición específica dentro de su rango de operación y son muy utilizados en sistemas de radio control, robótica y equipos industriales. (Yamid Ramirez, 2010)

Un servomotor está compuesto por un motor DC, una caja de engranajes reductora, un circuito de control y un sensor de velocidad y posición como se muestra en la Figura 2.8.

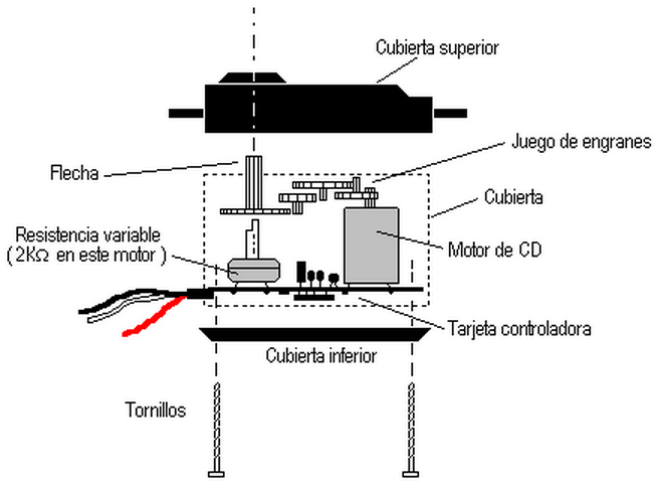


Figura 2.8 Partes de un Servomotor

Los servomotores están conectados por 3 terminales según los colores de los cables de conexión:

- Rojo: +Vcc (Tensión positiva) de 4 a 8 Voltios de Corriente Continua
- Negro: -Vcc (Potencial de referencia o borne negativo) 0 Voltios
- Blanco/Amarillo/Naranja: (Señal de control)

2.3.1. SERVOMOTOR TOWERPRO MG996

El servomotor que se muestra en la Figura 2.9 es ideal para todo tipo de proyectos electrónicos debido a que soporta un peso de hasta 15Kg. Entre las especificaciones del servomotor Towerpro M996 tenemos:

Figura 2.9 Servomotor Towerpro MG996

- Dimensiones: 40.7 x 19.7 x 42.9 mm
- Peso: 55gr
- Torque: 9 kg/cm (4.8V), 11 kg/cm (6V)
- Velocidad de operación:

  0.17sec/60 degree (4.8v)

  0.13sec/60 degree (6v)

- Voltaje de operación: 4.8-7.2 V
- Temperatura de operación: 0℃ - 55℃
- Engranes: metálicos

2.3.1.1. FUNCIONAMIENTO

A través del ancho de pulso de la señal cuadrada de voltaje (PWM) se fija la posición angular en que se encuentra el servomotor (como se muestra en la Figura 2.10).

Para el cálculo del periodo de tiempo del pulso se utilizó la siguiente expresión:

$$t = 1 + \frac{\emptyset}{180°}[ms]$$

Donde ϕ toma valor de 0° y 180°

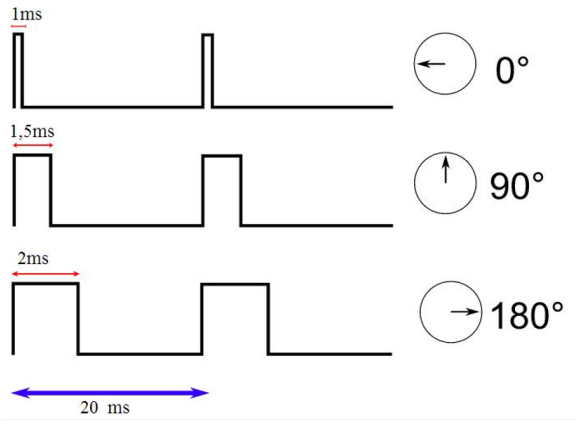


Figura 2.10 Duración de Ancho de Pulso según posiciones angulares

2.3.2. PWM

La modulación consiste en generar ondas cuadradas con una frecuencia y ciclo de trabajo determinado.

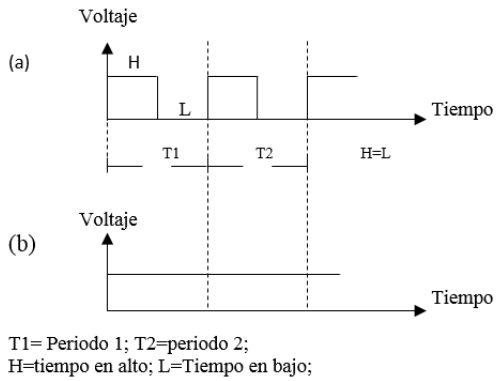


Figura 2.11 PWM Porcentajes de Comparación (a)50% (b)100%

2.4. TECNOLOGÍA BLUETOOTH

El estándar Bluetooth surgió en 1994, fue desarrollado por la empresa Sueca Ericsson que buscaba eliminar el cableado de las redes existentes. Es muy importante para establecer enlaces de comunicación inalámbrica para la transmisión de voz y datos entre dispositivos (ver Figura 2.12) mediante un enlace de radio frecuencia. Los dispositivos que utilizan esta tecnología son: Impresoras, teléfonos, consolas, teclados, altavoces, etc.

Su alcance es limitado y se lo utiliza para intercambiar datos entre dispositivos móviles a una corta distancia sin necesidad de cables.

Figura 2.12 Dispositivos con Tecnología Bluetooth

La tecnología Bluetooth trabaja en la banda mundial no licenciada ISM de 2.4 GHz, que corresponde a las bandas de UHF que ocupa desde los 300 MHz a los 3 GHz (ver Figura 2.13).

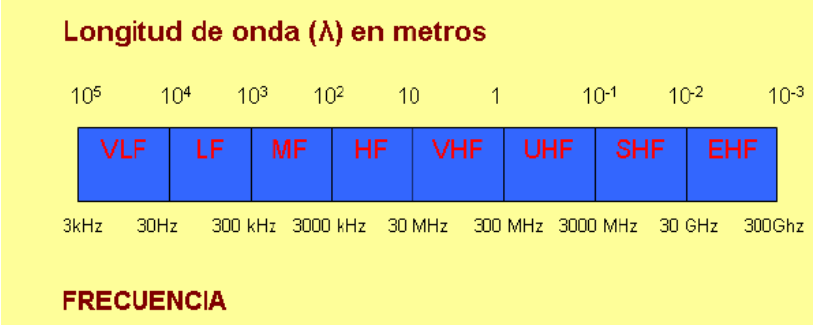


Figura 2.13 Ubicación del Espectro de Radio Frecuencia

Los estándares de Bluetooth están en el rango desde los 2.4 a 2.48 GHz, al trabajar en una banda no licenciada ISM y no pagar el uso de espectro electromagnético ésta tecnología puede ser considerada muy económica (ver Figura 2.14). (Raúl Hernández Aquino, 2008)

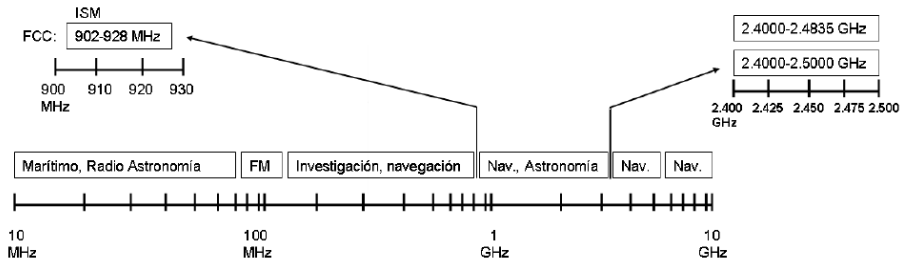


Figura 2.14 Ubicación de la Banda ISM de 2.4 GHz

2.4.1. CARACTERÍSTICAS

- El Bluetooth está orientada a la trasmisión de señales de voz y datos.
- El ancho de banda de frecuencia de 2.4 GHz y que no requiere de licencia.
- Posee una máxima velocidad de transmisión de 3 Mbps.
- El radio de alcance es de 1 a 100 metros dependiendo de la clase del dispositivo Bluetooth.

Un dispositivo maestro de dedica a la transmisión, mientras que in dispositivo esclavo adopta la posición de solamente recibir datos y voz.

2.4.2. MÓDULO DE COMUNICACIÓN BLUETOOTH HC06

El modulo que se muestra en la Figura 2.15 puede trabajar como maestro si solo se dedicara a la transmisión y como esclavo solamente si permite recibir datos y voz.

Figura 2.15 Módulo de Comunicación Bluetooth HC06

Los pines de control y alimentación del dispositivo HC06 están dados en la siguiente tabla.

PIN	SEÑAL
PIN_1	GND
PIN_2	DTX
PIN_3	DRX
PIN_4	+5VDC

Tabla 2.2 Pines del Módulo de Bluetooth

Los parámetros de configuración del fabricante están presentados en la siguiente tabla.

CONFIGURACIÓN INICIAL	
BAUD RATE	9600
PARITY BIT	NONE
DATA BIT	8
STOP BIT	1

Tabla 2.3 Configuración de fábrica del dispositivo Bluetooth HC06

Clase	Potencia máxima permitida (mW)	Potencia máxima permitida (dBm)	Alcance (aproximado)
Clase 1	100 mW	20 dBm	~100 metros
Clase 2	2.5 mW	4 dBm	~10 metros
Clase 3	1 mW	0 dBm	~1 metro

Tabla 2.4 Consumo de Potencia por clase de Bluetooth

Versión	Ancho de banda
Versión 1.2	1 Mbit/s
Versión 2.0 + EDR	3 Mbit/s
Versión 3.0 + HS	24 Mbit/s
Versión 4.0	24 Mbit/s

Tabla 2.5 Versiones de Tecnología Bluetooth

2.4.2.1. VENTAJAS

- No se necesita internet para la transmisión de datos y voz.
- Permite la comunicación con periféricos sin necesidad de cables.

2.4.2.2. DESVENTAJAS

- El rango máximo de transmisión es de 100 metros.
- Velocidad de transmisión muy lenta.
- El límite máximo de Interconexión es de hasta 7 periféricos.
- Alto consumo de potencia mientras el dispositivo se encuentra visible.
- Vulnerable a entrada de software malicioso.

2.5. DISPOSITIVO ULTRASÓNICO HCSR04

Es un dispositivo medidor de distancia que procesa pulso ultrasónico, este emite un sonido por el pin Trigger, el cual activa una señal al pin Echo y mide el tiempo que la señal tardo en regresar al Echo.

Figura 2.16 Sensor Ultrasónico HCSR04

Como se puede apreciar en la Figura 2.16 el dispositivo cuenta con pines de conexión los cuales son detallados de la siguiente manera:

- VCC = +5VDC
- Trig = Trigger input of Sensor
- Echo = Echo output of Sensor
- GND = GND

2.5.1. ESPECIFICACIONES DEL SENSOR HC-SR04

DATOS DE FÁBRICA	
Fuente de Alimentación	5 V
Corriente en reposo	2 mA
Corriente de trabajo	15 mA
Ángulo emisión de pulso	15°
Distancias de lectura	2cm - 400cm
Tiempo de respuesta	10 uS
Dimensión	45mm x 20mm x 15mm
Frecuencia	40 kHz

Tabla 2.6 Especificaciones del HC-SR04

El sensor de distancia ultrasónico HC–SR04 tiene la capacidad de medir la distancia mediante la diferencia de tiempo entre la transmisión y recepción de pulsos que el módulo envía. Para calcular la distancia en el dispositivo se utiliza la siguiente fórmula. (Soria, 2013)

$$\mathbf{Distancia} = (\mathbf{Ancho\ del\ pulso} * \mathbf{170})$$

2.6. ANDROID

2.6.1. DEFINICIÓN

Android es un sistema operativo basado en Linux, que se lo utiliza en dispositivos móviles como son los teléfonos inteligentes, tabletas y otros dispositivos.

Actualmente Android es la plataforma que lidera el market share (cantidad de dispositivos con Android), por encima de iOS, Symbian OS, RIM OS y Windows Mobile.

2.6.2. APPINVENTOR

App inventor es un framework creado inicialmente por el MIT (Instituto tecnológico de *Massachusetts*), para que cualquier usuario pueda crear su propia aplicación para teléfonos móviles con sistema operativo Android.

El entorno de desarrollo de App Inventor también es compatible con Mac OS X, GNU / Linux y sistemas operativos de Windows. En la Figura 2.17 se muestra los elementos principales de APPINVETOR.

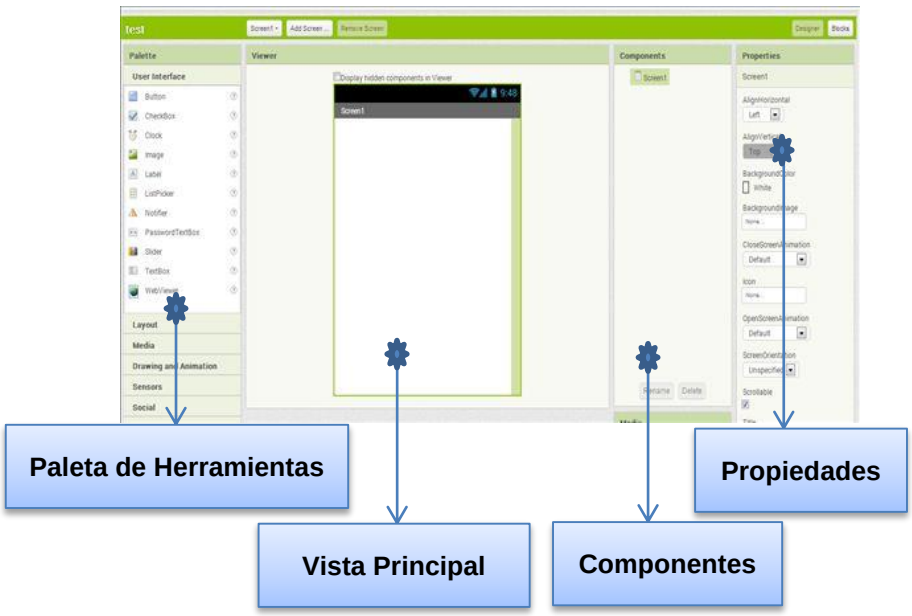


Figura 2.17 Elementos en APPINVENTOR

CAPÍTULO 3

3. DISEÑO E IMPLEMENTACIÓN

En este capítulo se presenta el diseño e implementación del proyecto, en la Figura 3.1 se muestra el diagrama de bloques del sistema y la interconexión de los componentes.

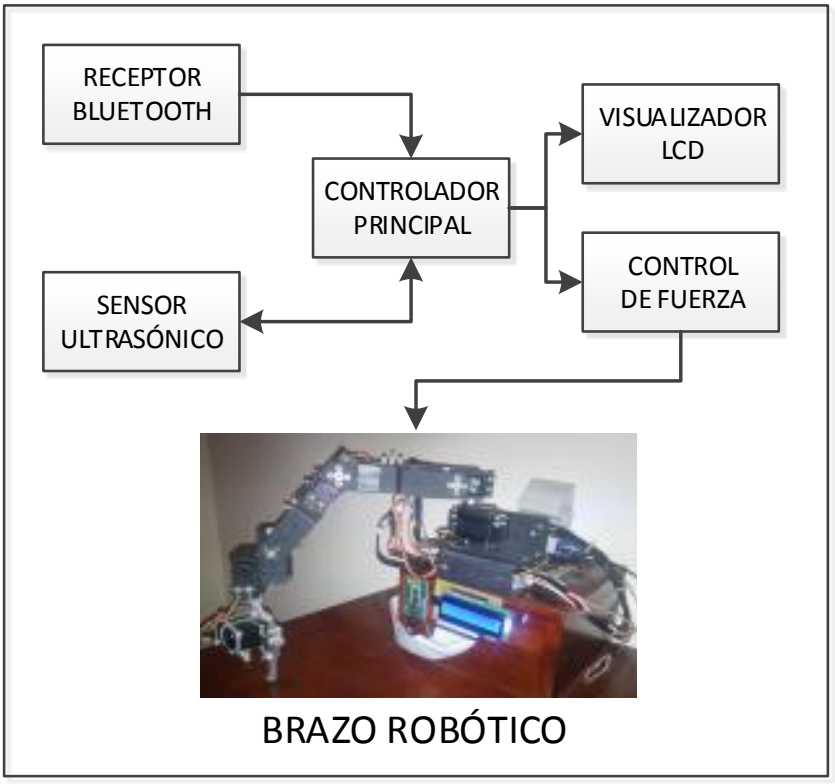


Figura 3.1 Diagrama de Bloques del Sistema

3.1. RECEPTOR BLUETOOTH

Para controlar remotamente el brazo robótico usamos el modulo Bluetooth HC-06 que permite recibir instrucciones desde un dispositivo móvil previamente enlazado con la aplicación de APPINVENTOR.

Parte del Brazo	Comando	ASCII	Hexadecimal	Binario
HOMBRO	BAJA	A	0X41	01000001
	SUBE	B	0X42	01000010
CODO	BAJA	C	0X43	01000011
	SUBE	D	0X44	01000100
MUÑECA	BAJA	E	0X45	01000101
	SUBE	F	0X46	01000110
BASE	IZQUIERDA	G	0X47	01000111
	DERECHA	H	0X48	01001000
MUÑECA GIRATORIA	IZQUIERDA	I	0X49	01001001
	DERECHA	J	0X4A	01001010
TENAZA	CERRAR	K	0X4B	01001011
	ABRIR	L	0X4C	01001100

Tabla 3.1 Comandos de control de App Inventor en modo automático

Estos caracteres son enviados por la aplicación enlazada al brazo robótico a través del Módulo Bluetooth y son interpretados procesador del sistema de control.

Para poder establecer un enlace de comunicación deberán por primera y única vez usar una clave de asociación, la cual viene configurada por defecto en el módulo (1234) y de esta manera nuestro dispositivo móvil funcionara como Master en la

comunicación ya que cada vez que se encuentre activo y disponible nuestro módulo HC-06 (Slave) se vinculara automáticamente y se podrá establecer la comunicación requerida.

3.2. SENSOR ULTRASÓNICO

Es usado cada vez que el brazo robótico se posiciona en alguna zona de su campo de acción, de manera que el procesador envía una señal ultrasónica con la ayuda del sensor y calcula el tiempo que demora en retornar dicha señal mediante la lectura del pin ECHO para calcular la distancia de la superficie cubierta por la tenaza.

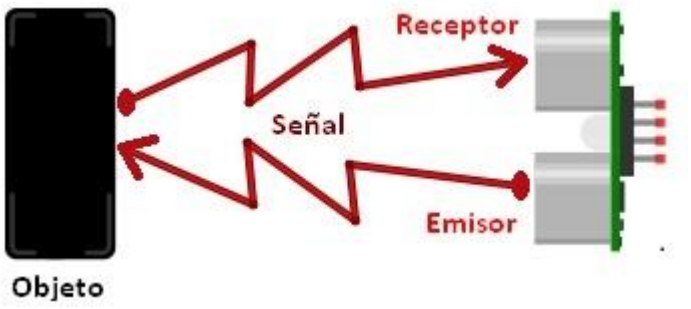


Figura 3.2 Envío de una Señal para detección de objetos.

Como se puede apreciar en la Figura 3.2 la señal ultrasónica enviada por el procesador con la ayuda del sensor viaja por el medio, en este caso el aire desde el emisor hasta rebotar en el objeto, esta se refleja y es detectada por el receptor.

Durante todo tiempo que la señal se mantuvo viajando en el medio el sensor mantiene en nivel lógico alto su PIN ECHO con lo cual se puede calcular la distancia a la que se encuentra el objeto.

3.3. VISUALIZADOR LCD

Es una pantalla LCD de 2x16 (Filas x Columnas) en la que se puede visualizar el modo del funcionamiento del brazo robótico y muestra si se detecta algún objeto entre las pinzas del brazo.

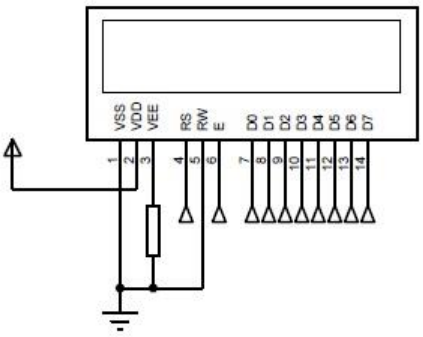


Figura 3.3 Conexión básica para escribir en la LCD

Puesto que solo necesitamos escribir en pantalla debemos conectar RW (PIN 5) de la LCD a GND como se muestra en la Figura 3.3.

Los pines RS y E son conectados directamente con el procesador ya que estos son los encargados de enviar los pulsos necesarios para procesar los comandos y los datos enviados a la LCD mediante los pines D0 - D7, siendo D7 el bit más significativo y D0 el menos significativo del código hexadecimal de la configuración o a su vez el código ASCII del dato enviado (carácter).

3.4. CONTROL DE FUERZA

Debido a que el nivel de voltaje de la señal de salida de la tarjeta DE0-Nano entrega señales con nivel de 3.3V, fue necesario acoplar un control de fuerza para convertir estas señales a 5V adecuadas para ser leídas para ser leídas por los pines de control de los servomotores.

3.4.1. ADAPTACIÓN DE NIVEL DE VOLTAJE

Para realizar la conversión de los niveles de voltaje para el control de los servomotores se cuenta con un circuito electrónico conformado por resistencias y optoacopladores, los cuales han sido interconectados a manera de switch tanto para la señal de entrada como para la señal de salida.

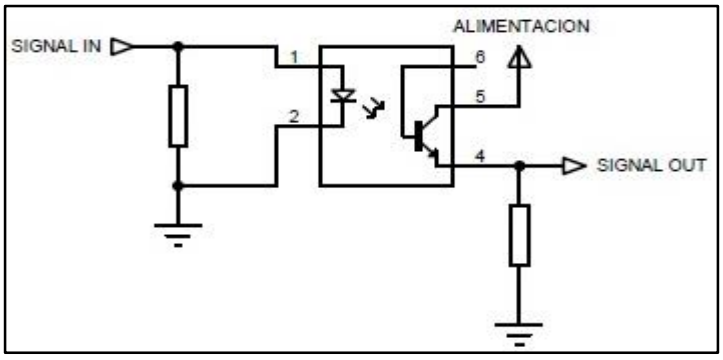


Figura 3.4 Convertidor de señales 3.3 Vcc a 5 Vcc

Como se observa en la Figura 3.4, al momento de aplicar una señal a la entrada de la configuración de nuestro adaptador de voltaje, el led interno del optoacoplador está polarizado con la misma tensión aplicada a la resistencia externa, luego esto permite que a la salida se conmute la fuente de alimentación con la resistencia ubicada en el emisor a la salida del optoacoplador y así nuestra señal de salida tendrá el mismo valor de voltaje que la alimentación proporcionada.

De otro modo si no se proporciona ninguna tensión a la entrada de nuestro adaptador de voltaje de señales la salida se mantendrá desactivada y con un valor de voltaje de 0V.

3.5. CREACIÓN DE LA COMPUTADORA BÁSICA USANDO LA HERRAMIENTA QSYS

La herramienta Qsys presenta una librería de componentes que nos permite añadir de acuerdo a los requerimientos básicos que debe poseer la mini-computadora para el control del brazo robótico.

Los componentes básicos requeridos para un correcto funcionamiento de nuestra mini-computadora son:

- **System ID Peripheral**: Es uno de los periféricos más importantes que debe contener el sistema, ya que permite a la herramienta de software de NIOS II saber si la aplicación desarrollada pertenece al hardware previamente configurado. Es esencial que solo exista un solo periférico de este tipo en cada sistema, en este caso lo nombramos “sysid”.

- **NIOS II Processor**: Encargado de ser el cerebro para todo el sistema, ya que todo el software será interpretado por él y este a su vez enviara las señales necesarias y correspondientes a

cada periférico para su correcto funcionamiento. En este proyecto es nombrado como “CPU”.

- **On-Chip Memory**: Esta es una memoria que se implementará en la FPGA con la cual almacenaremos el software del sistema y también todos los datos necesarios en el programa. Es nuestro sistema es nombrada como “Onchip_memory”.

- **JTAG UART**: Este periférico nos permite establecer la comunicación, es esencial incluir este módulo ya que de no ser así no se podría realizar ningún cambio a la configuración del chip Cyclone IV.

- **SDRAM Controller**: Este módulo nos ayuda a acceder y controlar la memoria SDRAM externa incluida en la tarjeta DE0-Nano, la cual tiene la capacidad de almacenar 32Mbytes de datos. En este proyecto se nombra como “SDRAM”.

- **Interval Timer**: Este módulo nos ayudara a medir intervalos de tiempo necesarios para nuestro programa. En este proyecto lo nombramos “Interval_Timer”.

- **UART (RS-232 Serial Port)**: Este módulo es de gran ayuda para establecer un enlace de comunicación serial entre 2 dispositivos. En este proyecto se establece la comunicación del sistema mediante el uso del módulo serial Bluetooth con el dispositivo inteligente (Tablet o Smartphone).

- **PIO (Parallel I/O)**: Es de gran importancia ya que es mediante este módulo que se obtienen los datos del exterior para poder ser procesados. De igual manera se entregan datos al exterior para el control de otros dispositivos acoplados al sistema o proporcionar información útil al usuario.

  En este proyecto se han usado varios módulos PIO y se han configurado con los siguientes propósitos:

  - Interacción con el sensor ultrasónico

    PIN_ECHO (entrada de 1bit).

    PIN_TRIGGER (salida de 1 bit).
  - Interacción con visualizador LCD

    EN_LCD (Salida de 1 bit).

    RS_LCD (Salida de 1 bit).

    DATA_LCD (Salida de 8 bits).

- **PWM:** Este módulo es el encargado de generar y entregar señales moduladas por ancho de pulso para el control de los servomotores. Se debe agregar el módulo a las librerías de Qsys.

Una vez agregado cada uno de los módulos anteriormente descritos y realizar la interconexiones entre los diferentes módulos tendremos nuestro sistema sin mensajes de error como se muestra en la Figura 3.5.

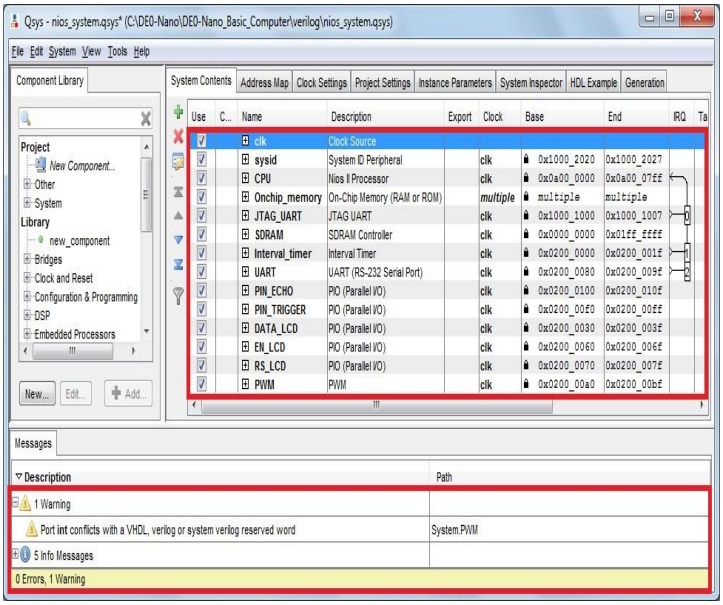


Figura 3.5 Módulos necesarios para el sistema creado en Qsys

3.5.1. VISUALIZACIÓN Y ASIGNACIÓN DE PINES A LA COMPUTADORA BÁSICA.

Luego de haber generado el hardware básico para el funcionamiento de nuestro proyecto y cerrar la ventana de Qsys. Podemos abrir un nuevo archivo esquemático de Quartus II y visualizar nuestro sistema creado en un solo bloque como se muestra en la Figura 3.6.

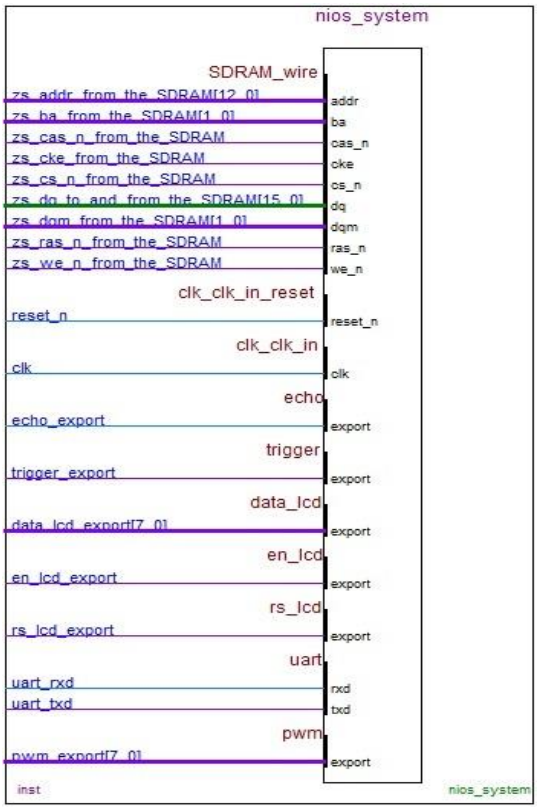


Figura 3.6 Representación en bloque del sistema creado en Qsys.

Siguiendo con la implementación de nuestro proyecto debemos asignar los pines necesarios para de esta manera poder recibir o entregar información a cada uno de los periféricos que conforman el sistema. Iniciamos con la asignación de pines del módulo controlador de la memoria SDRAM externa de acuerdo a lo proporcionado en la Tabla 3.2

SDRAM			
Node Name	**Location**	**Node Name**	**Location**
DRAM_ADDR[11]	PIN_N1	DRAM_DQ[15]	PIN_K1
DRAM_ADDR[10]	PIN_N2	DRAM_DQ[14]	PIN_N3
DRAM_ADDR[9]	PIN_P1	DRAM_DQ[13]	PIN_P3
DRAM_ADDR[8]	PIN_R1	DRAM_DQ[12]	PIN_R5
DRAM_ADDR[7]	PIN_T6	DRAM_DQ[11]	PIN_R3
DRAM_ADDR[6]	PIN_N8	DRAM_DQ[10]	PIN_T3
DRAM_ADDR[5]	PIN_T7	DRAM_DQ[9]	PIN_T2
DRAM_ADDR[4]	PIN_P8	DRAM_DQ[8]	PIN_T4
DRAM_ADDR[3]	PIN_M8	DRAM_DQ[7]	PIN_R7
DRAM_ADDR[2]	PIN_N6	DRAM_DQ[6]	PIN_J1
DRAM_ADDR[1]	PIN_N5	DRAM_DQ[5]	PIN_J2
DRAM_ADDR[0]	PIN_P2	DRAM_DQ[4]	PIN_K2
DRAM_BA[1]	PIN_M6	DRAM_DQ[3]	PIN_K5
DRAM_BA[0]	PIN_M7	DRAM_DQ[2]	PIN_L8
DRAM_CAS_N	PIN_L1	DRAM_DQ[1]	PIN_G1
DRAM_CKE	PIN_L7	DRAM_DQ[0]	PIN_G2
DRAM_CLK	PIN_R4	DRAM_DQM[1]	PIN_T5
DRAM_CS_N	PIN_P6	DRAM_DQM[0]	PIN_R6
DRAM_RAS_N	PIN_L2	DRAM_WE_N	PIN_C2

Tabla 3.2 Configuración de pines para la Memoria SDRAM

Para poder usar la fuente de reloj de 50Mhz y proporcionar una señal de reset mediante una de las botoneras de la tarjeta de desarrollo DE0-Nano, usamos la siguiente configuración de pines (ver Tabla 3.3)

SEÑALES GLOBALES	
Node Name	**Location**
CLOCK_50	PIN_R8
RESET	PIN_J15

Tabla 3.3 Configuración de pines para las señales globales

Tomando en cuenta que nuestro sistema recibe y envía señales al sensor ultrasónico, debemos realizar la asignación de pines como se muestran en la Tabla 3.4.

SENSOR ULTRASÓNICO	
Node Name	**Location**
ECHO	PIN_B3
TRIGGER	PIN_B4

Tabla 3.4 Configuración de pines para el Módulo del Sensor Ultrasónico

Como también utilizamos en nuestro sistema un visualizador LCD debemos configurar los pines que se usan para enviar la información que se desea mostrar, como se aprecia en la Tabla 3.5.

VISUALIZADOR LCD	
Node Name	**Location**
DATA[7]	PIN_N16
DATA[6]	PIN_P16
DATA[5]	PIN_L15
DATA[4]	PIN_K16
DATA[3]	PIN_N11
DATA[2]	PIN_P9
DATA[1]	PIN_R10
DATA[0]	PIN_R11
EN	PIN_T15
RS	PIN_T13

Tabla 3.5 Configuración de pines para el Módulo LCD

Además necesitamos añadir los pines usados para establecer la comunicación serial (ver Tabla 3.6).

UART	
Node Name	**Location**
RX	PIN_T9
TX	PIN_F13

Tabla 3.6 Configuración de pines para el Módulo de Comunicación Serial

Finalmente recordando que lo primordial para el control de los servomotores que conforman nuestro brazo robótico es una señal modulada por ancho de pulsos, debemos asignar un pin a cada señal PWM necesaria como se muestra en la tabla 3.7.

PWM	
Node Name	**Location**
PWM[7]	PIN_E10
PWM[6]	PIN_B11
PWM[5]	PIN_D11
PWM[4]	PIN_B12
PWM[3]	PIN_D9
PWM[2]	PIN_E9
PWM[1]	PIN_F8
PWM[0]	--------

Tabla 3.7 Configuración de pines para el Módulo generador de señales PWM

3.6. CÓDIGO DEL PROGRAMA PRINCIPAL

Para el desarrollo de nuestro programa se utilizó las líneas de código presentes en los archivos Sk_Pwms.c y Sk_Pmws.h. Para hacer esto posible es necesario incluir el archivo de cabecera correspondiente a más de incluir las librerías necesarias para trabajar con cada uno de los componentes incluidos en el hardware (system.h) y usar los prototipos de funciones ya conocidos en Lenguaje C.

```
/******LIBRERIAS******/
#include "alt_types.h"
#include "Sk_Pwms.h"
#include <stdio.h>
#include <system.h>
#include  <string.h>
```

Figura 3.7 Librerías usadas en el desarrollo del Software del Sistema

Para el manejo de la pantalla LCD se crearon las siguientes funciones como se muestra en la Figura 3.8

```
/****MANEJO DEL VISUALIZADOR LCD****/
void Lcd_Init(){
    ProcesarComando(0x38);
    ProcesarComando(0x0C);
    ProcesarComando(0x06);
}
void Lcd_Clear(){
    ProcesarComando(0x01);
    usleep(1640);
}
void Pulso_EN(){
    IOWR(EN_LCD_BASE,0,1);
    usleep(1);
    IOWR(EN_LCD_BASE,0,0);
}
void ProcesarComando(char comando){
    volatile int *Dato= 0x2000030;
    *Dato=0x00;
    IOWR(RS_LCD_BASE,0,0);
    *Dato=comando;
    Pulso_EN();
    usleep(50);
}
void ProcesarDato(char letra){
    volatile int *Dato= 0x2000030;
    *Dato=0x00;
    IOWR(RS_LCD_BASE,0,1);
    *Dato=letra;
    Pulso_EN();
    usleep(50);
}
void Lcd_Write(char* cadena){
    int i=0;
    ProcesarComando(0x80);
    for(i=0; i<strlen(cadena); i++){
        if(i==16)
            ProcesarComando(0xC0);
        ProcesarDato(cadena[i]);
        usleep(12500);
    }
}
void Lcd_Erase(){
    int i=0;
    ProcesarComando(0x80);
    for(i=0; i<31; i++){
        if(i==16)
            ProcesarComando(0xC0);
        ProcesarDato(' ');
        usleep(12500);
    }
}
```

Figura 3.8 Funciones para el LCD

En estas funciones se especifican como se deben manejar los bit EN y RS además de la información entregada a través de la salida DATA de 8 bits para poder mostrar lo que deseamos en el visualizador LCD.

La primera función **Lcd_Init** se encarga de configurar el visualizador LCD para usar el bus de datos con una cantidad de 8 bit (DATA) y habilita ambas líneas del visualizador (Comando 0x38). Luego enciende el visualizador (Comando 0x0C) y lo configura de manera que en cada escritura de caracteres el cursor se autoincremente hacia la derecha (Comando 0x38).

La función **Lcd_Clear** se encarga de limpiar el contenido mostrado en el visualizador cada vez que se lo desee. La función **Pulso_EN** se encarga de enviar un pulso (flanco positivo) con duración de 1useg con el fin de que se cargue el dato o comando a ser procesado en el visualizador.

Las funciones **ProcesarComando** y **ProcesarDato** se encargan de cargar el valor del comando o dato a ser procesado en el segmento DATA poniendo el estado del pin RS en 0 y 1 respectivamente para diferenciar entre un comando o un dato, seguido de un pulso usando el pin EN para que este sea cargado.

Finalmente tenemos las funciones **Lcd_Write** la cual recibe como parámetro un texto cualquiera y lo escribe en el visualizador con la ayuda de las funciones mencionadas anteriormente, la función **Lcd_Clear** se encarga de borrar automáticamente carácter por carácter lo que se encuentre mostrando en determinado momento el visualizador LCD.

```
int mover(int valactual,int valfinal,int n_chanel){
    int i=valactual;
    while(i!=valfinal){
        usleep(20000);
        if(valfinal>valactual)
            i= i + 25;
        else
            i= i - 25;
        PWMS_SetDutyCycle(PWM_BASE,n_chanel,i);
    }
    return i;
}
```

Figura 3.9 Función para el Control de Servomotores

Para realizar el control de cada uno de los servomotores independientemente del sentido de giro, usamos la función **mover** (ver Figura 3.22) la cual recibe como parámetros la posición actual de los servomotores (Duty_Cycle) y la posición que se desea alcanzar, esta posición esta también referenciada con el Duty_Cycle, además de recibir el canal PWM que se usara para realizar la modificación del ancho de pulso a su salida.

Finalmente tenemos la función **distancia_tenaza** la cual con la ayuda del módulo Interval timer y los PIO pin_trigger y pin_echo calcula la distancia a la que se encuentra un objeto medido desde la tenaza del brazo robótico. Para esto inicializa y mantiene pausado el valor del timer, luego se procede a enviar un pulso con duración de 10useg a través del pin_trigger del sensor ultrasónico, una vez enviado el pulso se activa el timer para que empiece la cuenta.

```
float distancia_tenaza( int trigger_base, int echo_base){
    int contador_us=0;
    float distancia;
    long numbajo, numalto, numecloks;
    IOWR(INTERVAL_TIMER_BASE,2,0Xffff);
    IOWR(INTERVAL_TIMER_BASE,3,0xffff);
    IOWR(trigger_base,0,0);
    IOWR(trigger_base,0,1);
    usleep(10);
    IOWR(trigger_base,0,0);
    while(!IORD(echo_base,0));
    IOWR(INTERVAL_TIMER_BASE,1,4);
    while(IORD(echo_base,0));
    IOWR(INTERVAL_TIMER_BASE,1,8);
    usleep(50000);
    IOWR(INTERVAL_TIMER_BASE,4,0);
    IOWR(INTERVAL_TIMER_BASE,5,0);
    numbajo=0xffff & IORD(INTERVAL_TIMER_BASE,4);
    numalto=0xffff & IORD(INTERVAL_TIMER_BASE,5);
    numecloks=0xffffffff & (numbajo + (numalto<<16));
    contador_us=((0xffffffff-numecloks)/50);
    usleep(30000);
    distancia=contador_us/58.30;
    return distancia;
}
```

Figura 3.10 Función para calcular la distancia a la que se encuentra un objeto desde la Tenaza.

El timer se mantiene contando mientras que el estado lógico del pin_echo se mantenga en alto. Cuando el nivel lógico de este pin mencionado se ponga en bajo, capturamos el valor del timer y procedemos a realizar los cálculos respectivos para obtener así la distancia existente hasta el objeto apuntado por la tenaza (ver Figura 3.10).

3.6.1. DIAGRAMA DE FLUJO DEL PROGRAMA

Con la ayuda de las funciones descritas anteriormente procedemos a escribir un programa que funcione de acuerdo a los diagramas de flujo mostrados en las Figuras 3.11, 3.12, 3.13 y 3.14.

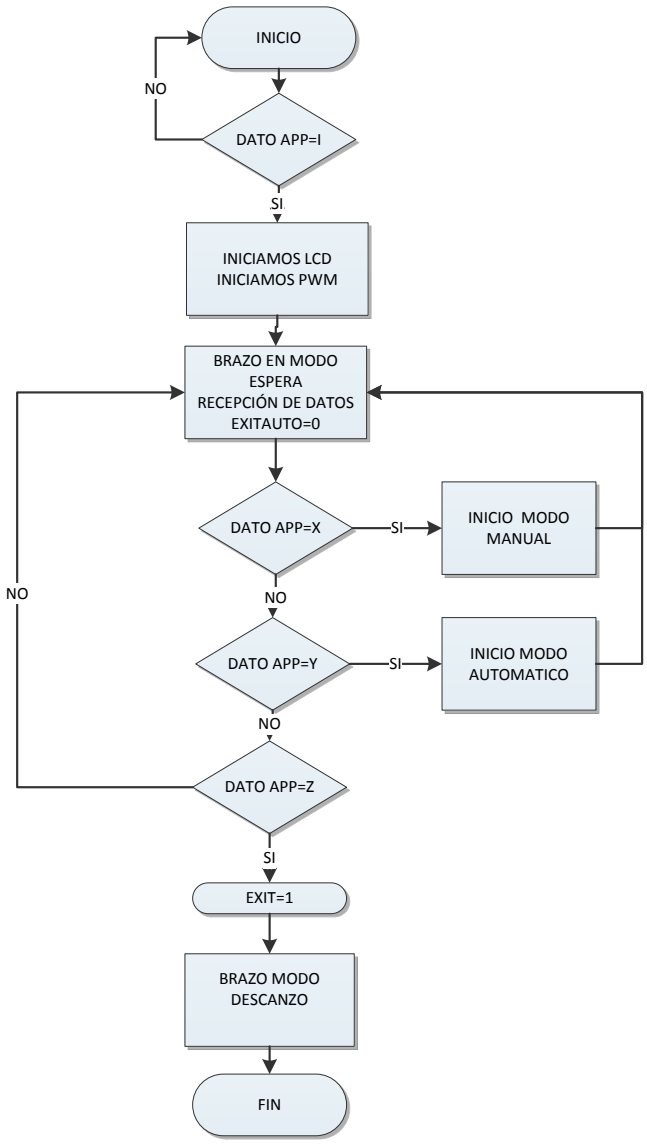


Figura 3.11 Diagrama de flujo del comportamiento general del Software del Sistema

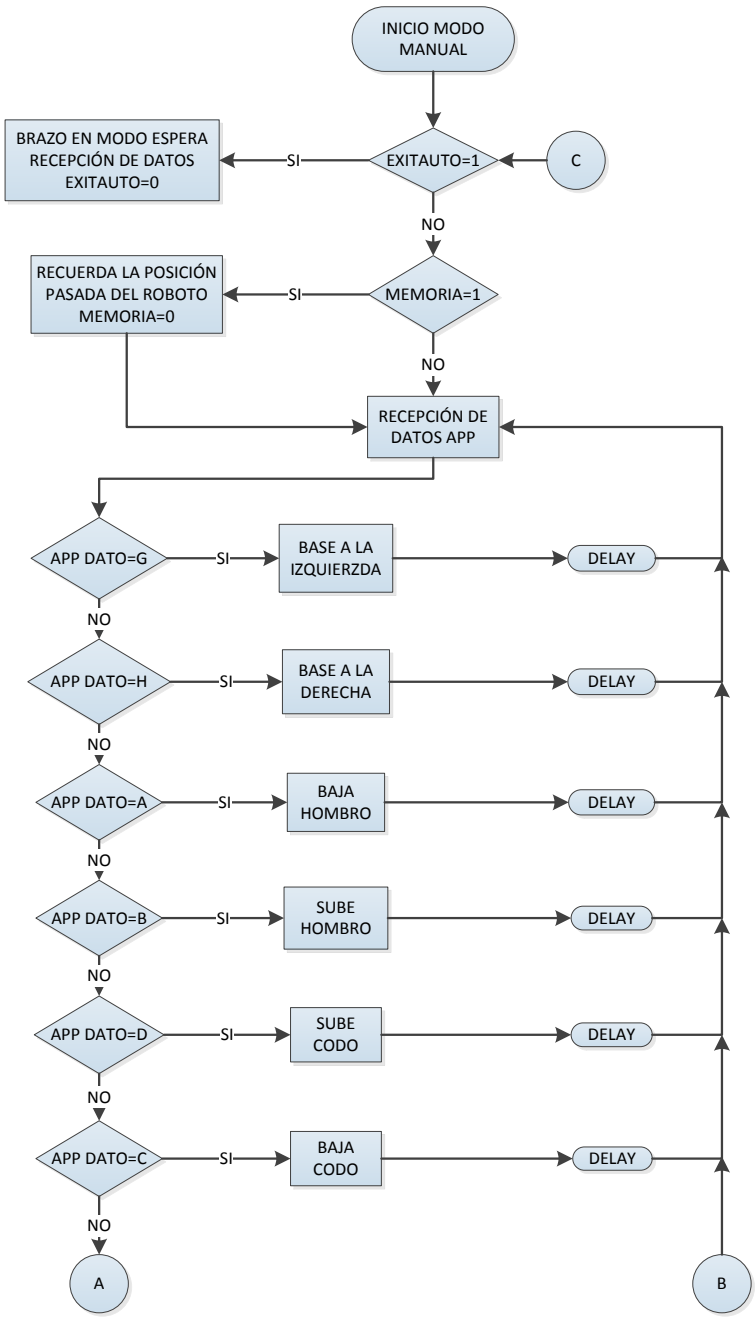


Figura 3.12 Diagrama de flujo del comportamiento del Modo Manual parte 1

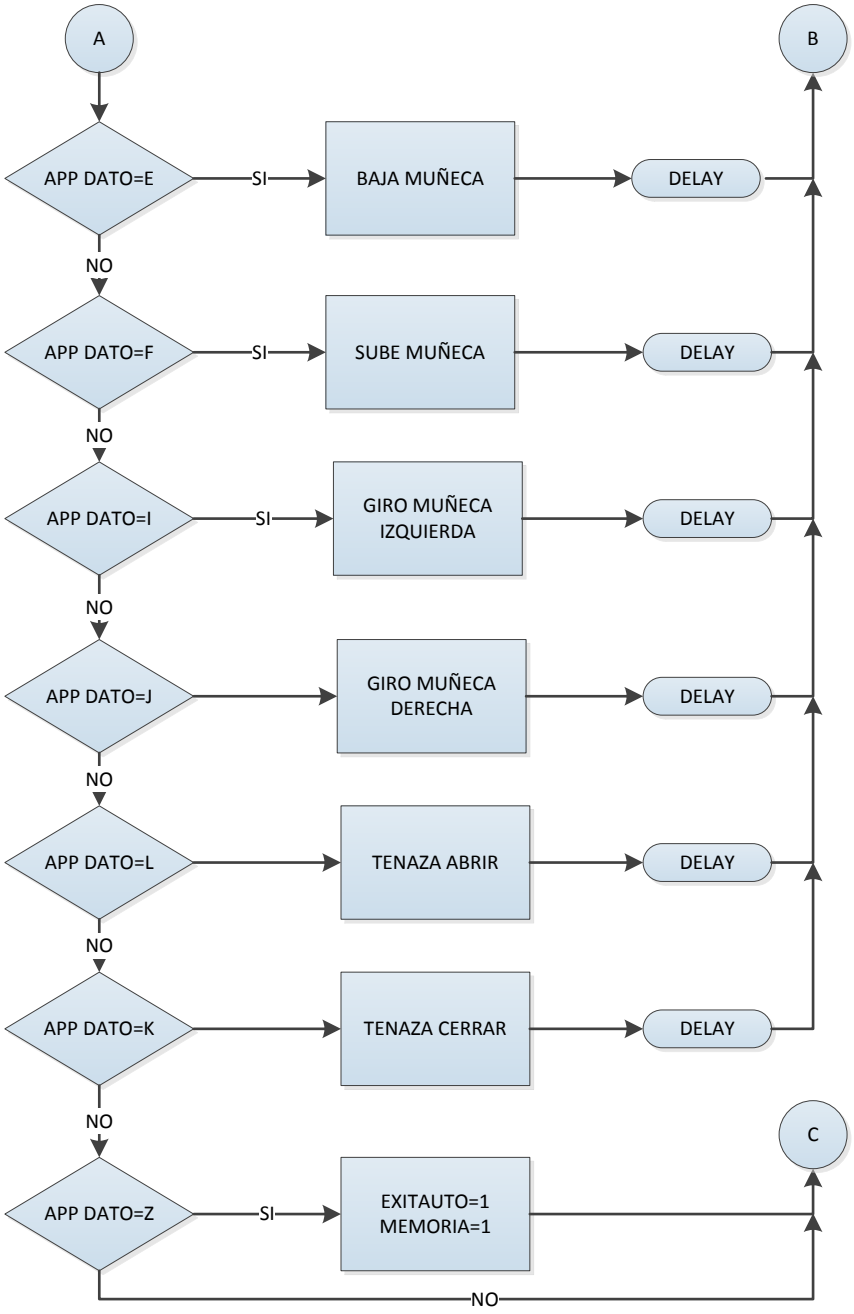


Figura 3.13 Diagrama de flujo del comportamiento del Modo Manual parte 2

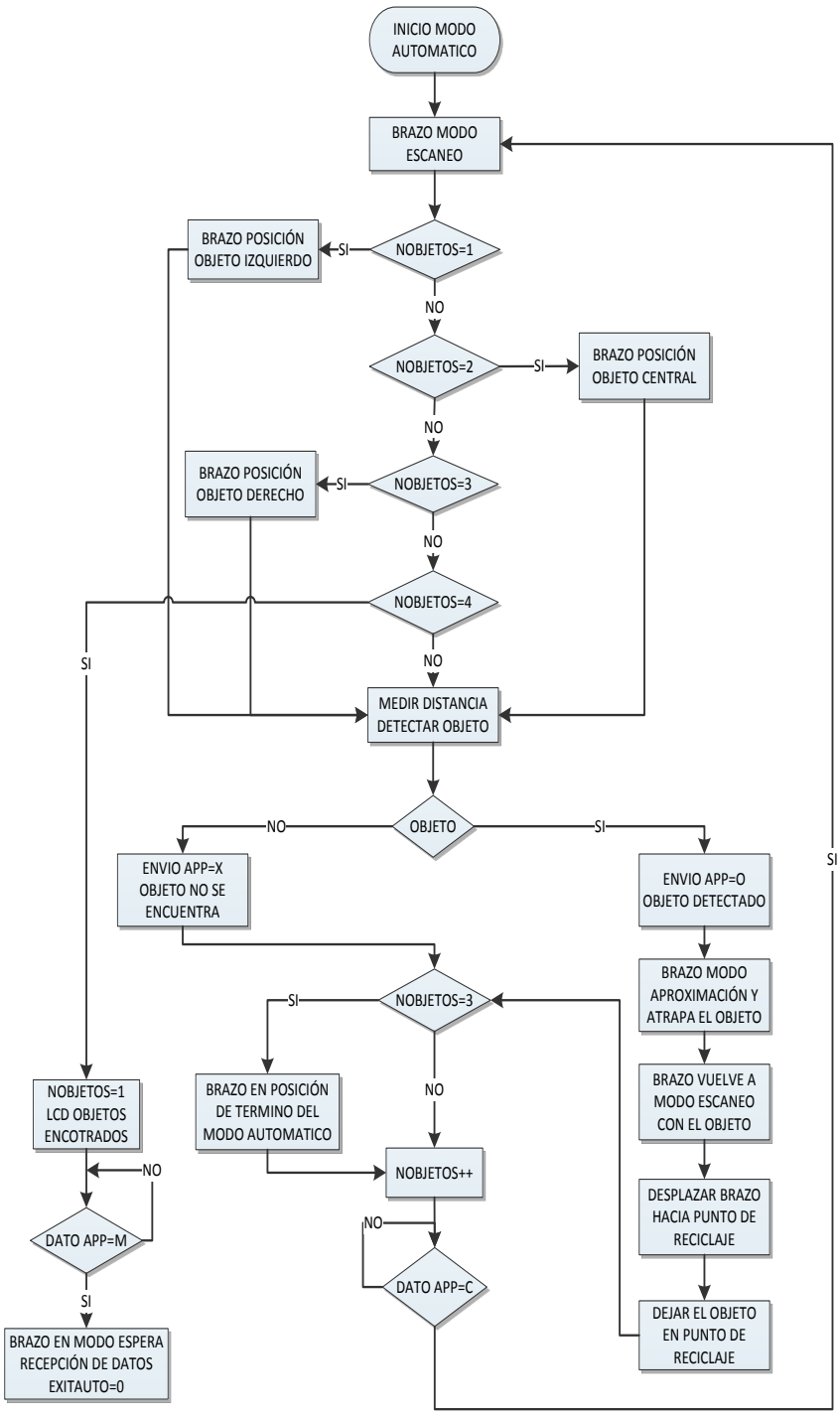


Figura 3.14 Diagrama de flujo del comportamiento del Modo Automático

3.7. PROGRAMACIÓN EN APPINVENTOR

Se diseña e implementa una aplicación capaz de gestionar el controlar un brazo robótico, para lo cual se utiliza la herramienta de desarrollo en su primera versión MIT App Inventor beta.

Para la programación en ésta plataforma, básicamente se crea la interfaz de usuario desde su página en internet que se encuentra con él mismo nombre y después mediante un editor de bloques interactivos se puede añadir la lógica a cada uno de ellos.

Se procede a crear la aplicación que trabaja sobre el sistema operativo Android. La Figura 3.15 define los bloques lógicos que hacen posible la visualización.

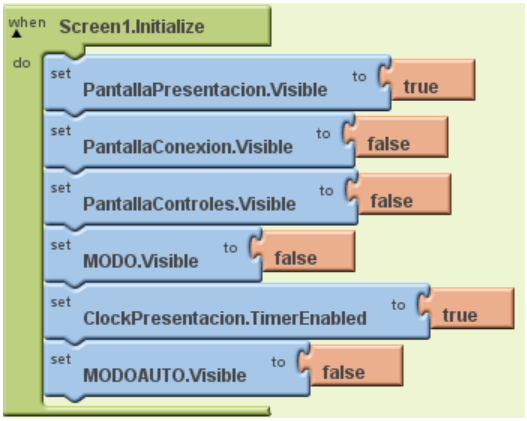


Figura 3.15 Bloques que dan inicio a la aplicación

En la Figura 3.16 se puede observar una pantalla de presentación que se mantiene activa por aproximadamente 6 segundos

Figura 3.16 Pantalla de presentación de la aplicación

Una vez pasados los 6 segundos se muestra una segunda pantalla, en la cual el operador puede elegir entre establecer un enlace de comunicación serial asíncrono vía Bluetooth con el botón **CONECTAR** o también puede optar por abandonar o parar la aplicación con el botón **CERRAR**. (Ver Figura 3.13).

Figura 3.17 Pantalla para establecer la comunicación Bluetooth entre la aplicación y el Brazo Robótico

Al presionar **CONECTAR**, antes de enlazarse con la dirección MAC del módulo Bluetooth HC-06, se pregunta si es posible establecer la conexión como se puede observar en la Figura 3.18 y luego si es posible se conecta con éste.

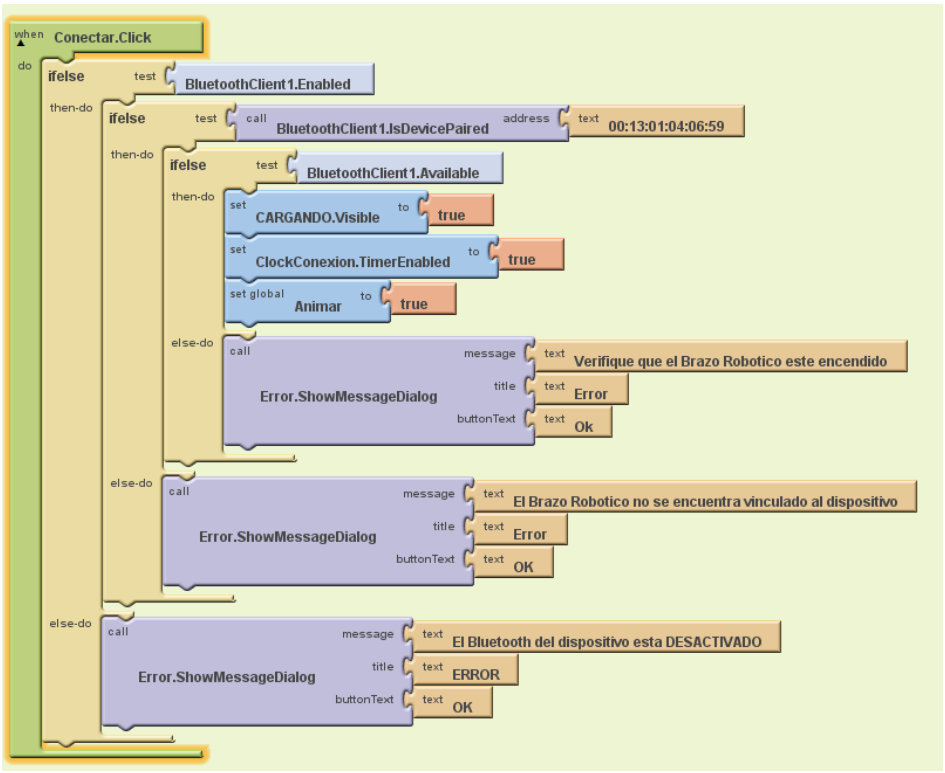


Figura 3.18 Arreglo de Bloques Lógicos que dan inicio a la conexión entre los dispositivos

Una vez realizada la conexión entre los dispositivos, entonces es posible el envío de tramas que son interpretados por nuestro sistema embebido de forma binaria.

Para que nuestro brazo robótico se inicialice, la aplicación envía el carácter **“I”** (ver Figura 3.19), que provoca que los servomotores coloquen a la estructura metálica en una posición de espera para la recepción de datos.

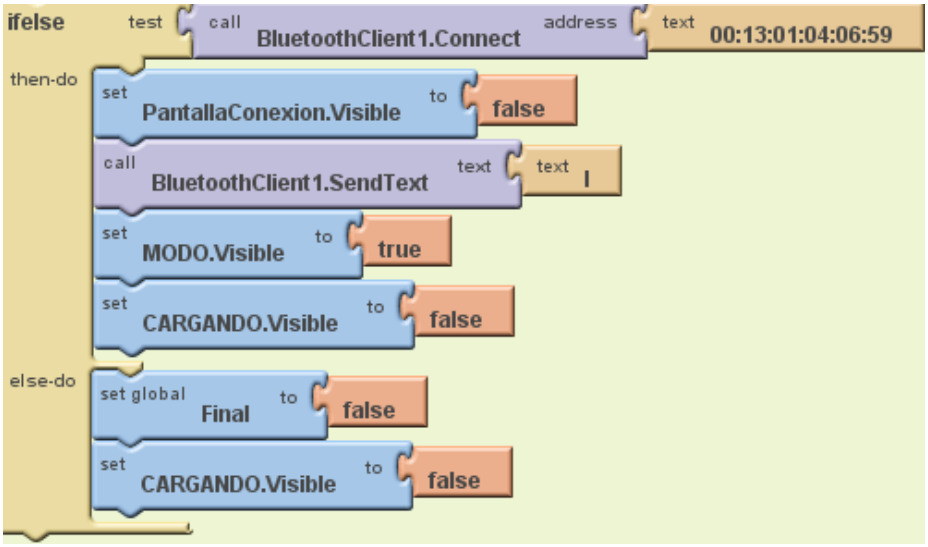


Figura 3.19 Arreglo de bloques que dan la condición de envío del carácter de inicio

Terminada la instrucción anterior, ahora se muestra una nueva pantalla, en la cual se puede elegir entre los siguientes modos de operación que tiene el brazo robótico, siendo éstos **MODO MANUAL**, **MODO AUTOMÁTICO** y de manera adicional encontramos opción **SALIR**.(ver Figura 3.20).

Figura 3.20 Pantalla de selección de Modo Manual o Automático

Al seleccionar el **MODO MANUAL**, se envía el carácter **"X"** al programa de control del brazo robótico y muestra una nueva pantalla que me permite la manipulación de cada uno de los servomotores por separado como se aprecia en la Figura 3.21.

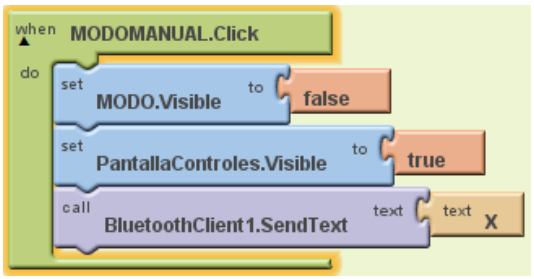


Figura 3.21 Bloque Lógico del botón MODO MANUAL

Por otro lado si presionamos **MODO AUTOMÁTICO**, se envía el carácter **"Y",** indicándole al programa del brazo robótico en qué modo tiene que a operar e inicia un conteo (set Check.TimerEnable.to true), en el que examina si el objeto es encontrado o no, repitiéndose el proceso hasta que el escaneo de objetos termine (ver Figura 3.22).

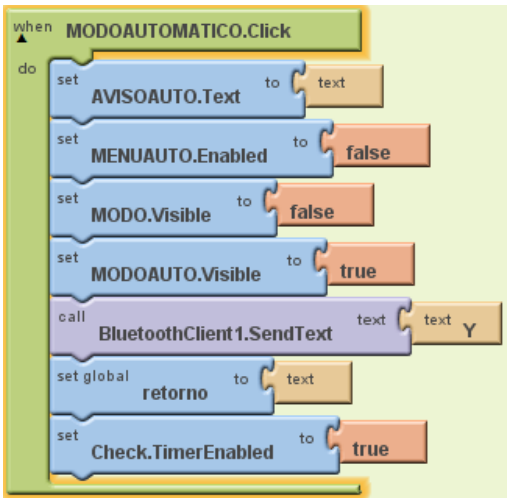


Figura 3.22 Bloque Lógico del botón MODO AUTOMÁTICO

En la Figura 3.23 se puede visualizar la interfaz de usuario en modo manual, en la cual el usuario o el operador del brazo robótico, elige cual parte mover. Adicional a esto cada vez que presionamos los botones de desplazamiento se envían caracteres, que son receptados e interpretados según la lógica programada en la tarjeta DE0-Nano.

Figura 3.23 Pantalla de Modo Manual

[1] Indica que parte del brazo robótico se desplaza.

[2] Selecciona la tenaza.

[3] Selecciona girar la tenaza.

[4] Selecciona la muñeca.

[5] Selecciona el codo.

[6] Selecciona el hombro.

[7] Selecciona la cintura.

[8] Dependiendo de que parte deseemos mover, desplaza a la izquierda la cintura.

[9] Dependiendo de que parte deseemos mover, desplaza a la derecha la cintura

[10] Regresa a la pantalla de selección de modos, guardando la última posición.

En la Figura 3.24 se puede visualizar la interfaz de usuario en modo automático, en la cual podemos observar figuras representativas de los objetos que si son detectados se pone de color verde de lo contrario se torna color rojo, además podemos observar un cuadro de texto que describe la acción de recolectar.

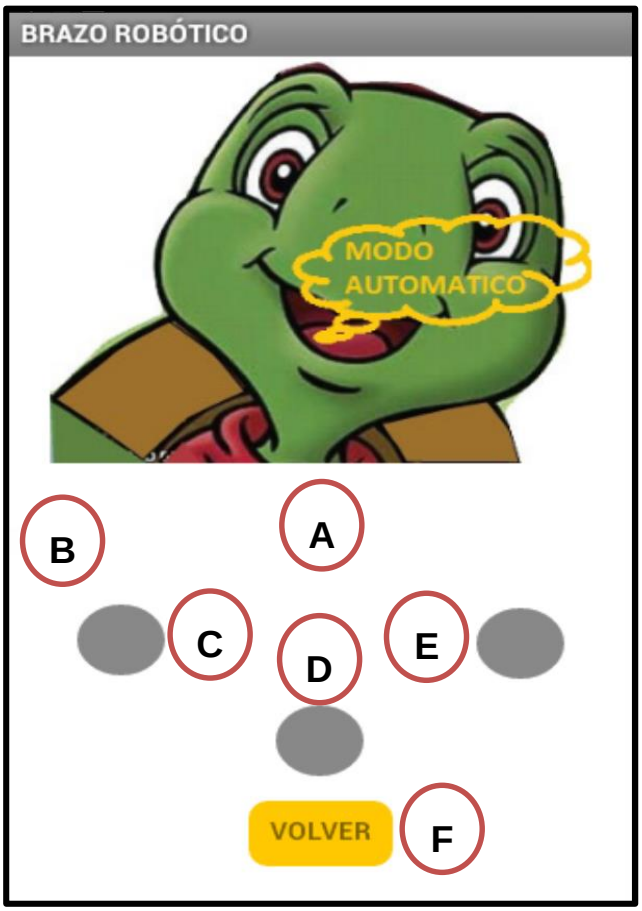


Figura 3.24 Pantalla de Modo Automático

[A] Muestra un mensaje si es que el objeto es detectado.

[B] Indica el número de objetos recolectados.

[C] Representa el objeto 1.

[D] Representa el objeto 2.

[E] Representa el objeto 3.

[F] Se activa una vez que termina el escaneo y regresa a la pantalla de selección de modos.

A continuación se muestra el diagrama de flujo que describe el funcionamiento de la aplicación que controla el brazo robótico (ver Figura 3.25).

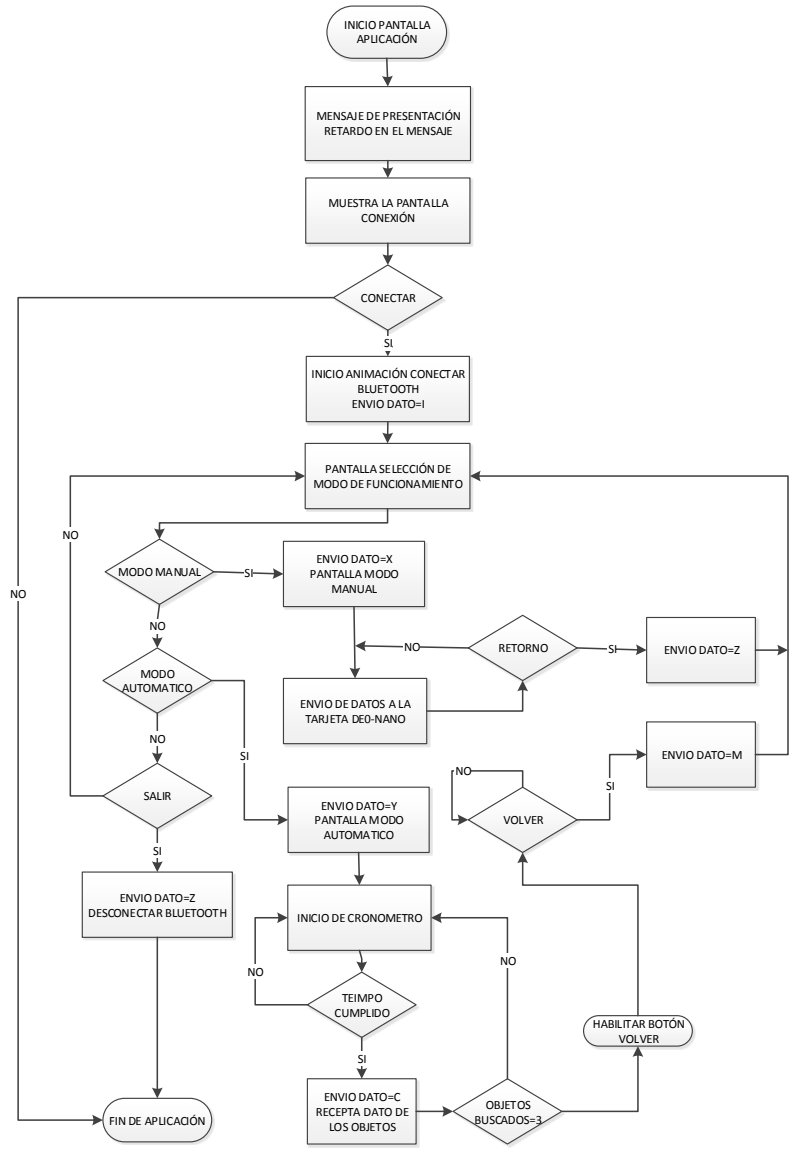


Figura 3.25 Diagrama de flujo de la aplicación en App Inventor

3.7.1. SIMULADOR DE APP INVENTOR.

La herramienta de desarrollo App Inventor, nos permite simular un dispositivo que trabaja con el sistema operativo de Android. Así con la ayuda de este simulador podemos testear la apariencia y funcionalidad de nuestro diseño.

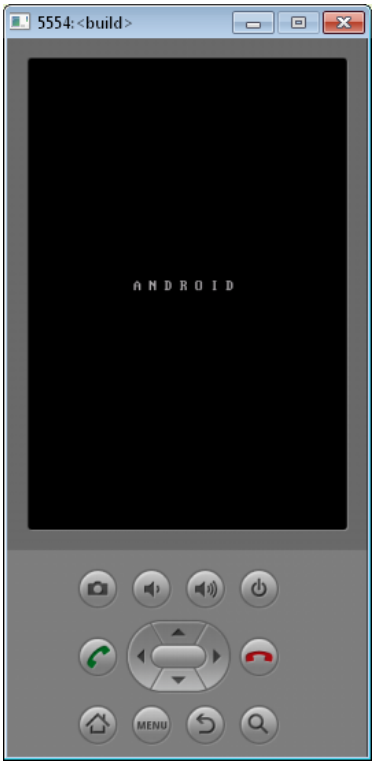


Figura 3.26 Simulador de App Inventor

A continuación presentamos algunas imágenes, de como se aprecia nuestra aplicación corriendo en el simulador de App Inventor.

Figura 3.27 Simulador de App Inventor pantalla presentación

Figura 3.28 Simulador de App Inventor pantalla conexión

3.8. MATERIALES Y COSTOS

Componente	Cantidad	Valor Unidad [$]	Valor total [$]
Tarjeta DE0 Nano	**1**	**120**	**120**
Tarjeta de acoplamiento (etapa de fuerza)	**1**	**15**	**15**
PCB para LCD y Bluetooth (interfaz de visualización)	**1**	**23**	**23**
Modulo Bluetooth	**1**	**40**	**40**
Carcasa Robótica	**1**	**280**	**280**
Sensor distancia	**1**	**15**	**15**
Servo Motores (Fuerza)	**2**	**40**	**80**
Servo Motores (Velocidad)	**4**	**14**	**56**
Tablet Android (Samsung Galaxy Tab 3)	**1**	**250**	**250**
Fuente de Alimentación	**1**	**15**	**15**
		TOTAL	**894**

Tabla 3.8 Lista de precios de componentes del proyecto

CAPÍTULO 4

4. PRUEBAS Y RESULTADOS

Este capítulo detalla los resultados de algunas pruebas y mediciones realizadas, con el fin de poder comparar los datos esperados del funcionamiento con los valores obtenidos en la implementación.

4.1. ANÁLISIS DE SEÑALES PWM PARA EL CONTROL DE SERVOMOTORES.

Debido a que cada señal de control de los servomotores pasa por nuestro circuito controlador de fuerza diseñado con optoacopladores (4N25), se realiza una comparación entre las señales de entrada y de salida.

Servomotor Cintura

- Modelo: hs-311
- Posición: 90°

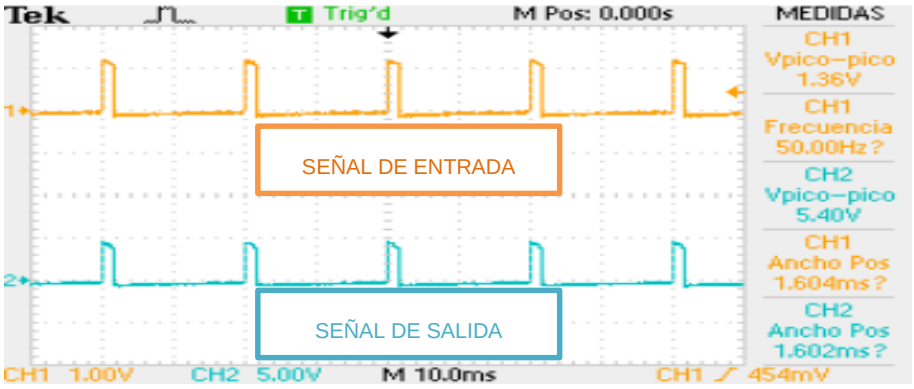


Figura 4.1 Señal PWM de control para Servomotor Cintura

Servomotor Hombro

- Modelo: TGY 1501
- Posición: 90°

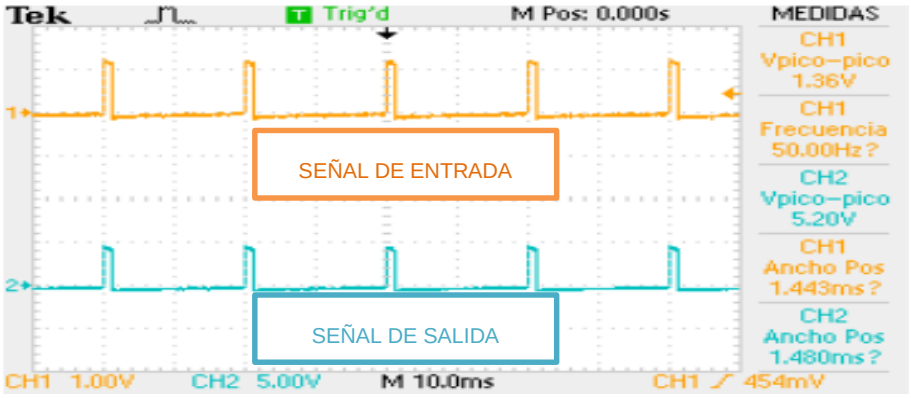


Figura 4.2 Señal PWM de control para Servomotor Hombro

Servomotor Codo

- Modelo: MG996R (TowerPro)
- Posición: 90°

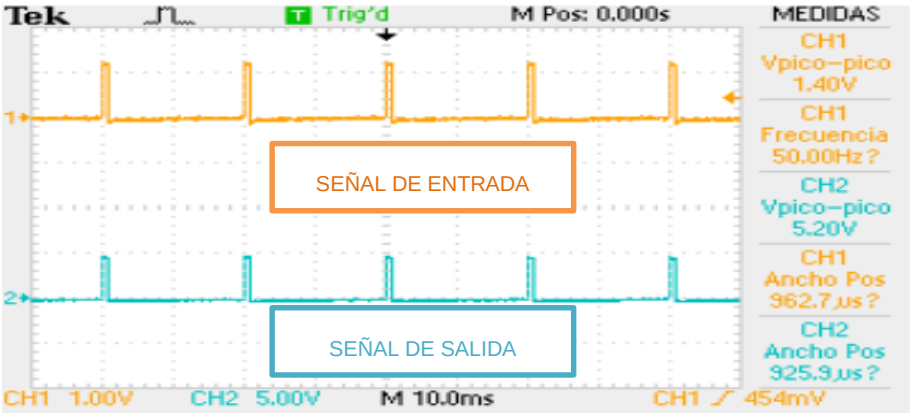


Figura 4.3 Señal PWM de control para Servomotor Codo

Servomotor Muñeca

- Modelo: 6001HB (PowerHd)
- Posición: 90°

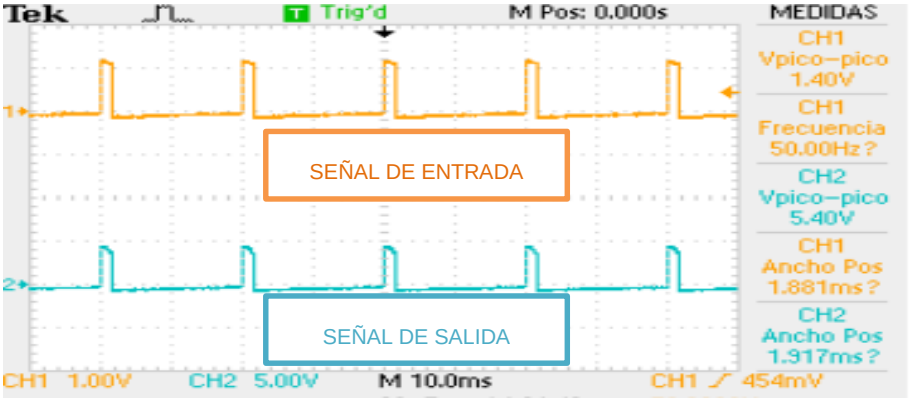


Figura 4.4 Señal PWM de Control para Servomotor Muñeca

Servomotor Muñeca Giratoria

- Modelo: 6001HB (PowerHd)
- Posición: 90°

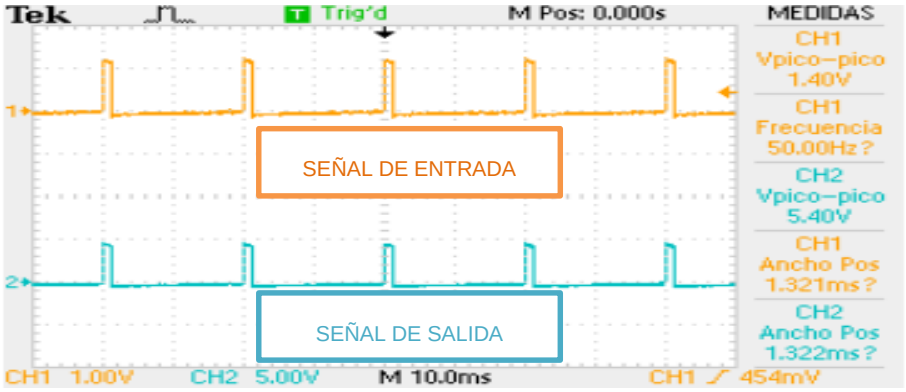


Figura 4.5 Señal PWM de Control para Servomotor Muñeca Giratoria

Servomotor Tenaza

- Modelo: Hs-311 (PowerHd)
- Posición: 180°

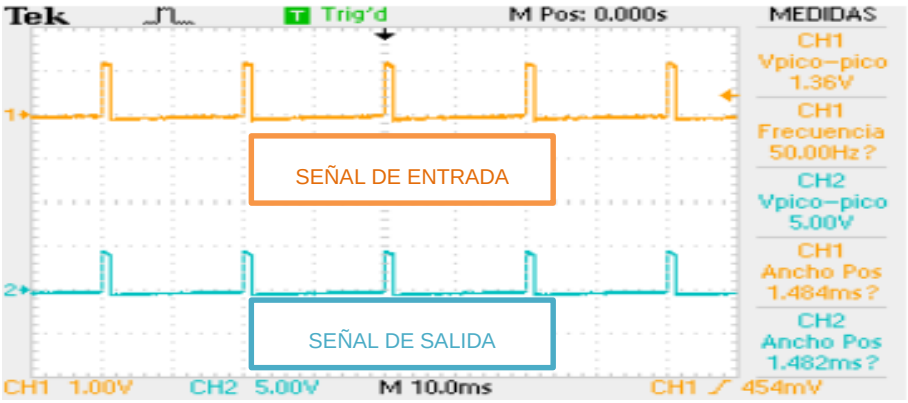


Figura 4.6 Señal PWM de Control para Servomotor Tenaza

En la Tabla 4.1 se muestra que el mayor porcentaje de error se da en las partes del hombro codo y muñeca.

PARTE	MODELO	SEÑAL ENTRADA [useg]	SEÑAS SALIDA [useg]	ERROR [%]
BASE	HS-311	1604	1602	**0,12%**
HOMBRO	TGY 1501	1443	1480	**2,56%**
CODO	MG996R	962,7	925,9	**3,82%**
MUÑECA	6001HB	1881	1917	**1,91%**
MUÑECA GIRATORIA	6001HB	1321	1322	**0,08%**
TENAZA	HS-311	1484	1482	**0,13%**

Tabla 4.1 Cálculo de error entre señales de entrada y salida del control de fuerza

4.2. ANÁLISIS DE DISTANCIA DE DETECCIÓN DE OBJETOS

En éste análisis se toma en cuenta cual es la distancia que el sensor ultrasónico HC-SR04 interpreta mediante nuestro algoritmo implementado, para así conocer los errores de medición. En la Tabla 4.2 tenemos los valores teóricos de distancia medidos desde la posición del sensor ultrasónico y el campo de acción en el que opera el brazo robótico.

	VALOR TEÓRICO	
SIN OBJETO	14,2	[cm]
CON OBJETO	11,7	[cm]

Tabla 4.2 Distancias Teóricas entre el sensor y el campo de acción

Se realizan cinco escaneos con objetos y cinco sin la presencia de ellos, para luego obtener un promedio de la distancia media en cada una de las posiciones fijadas para que el brazo robótico realice la búsqueda.

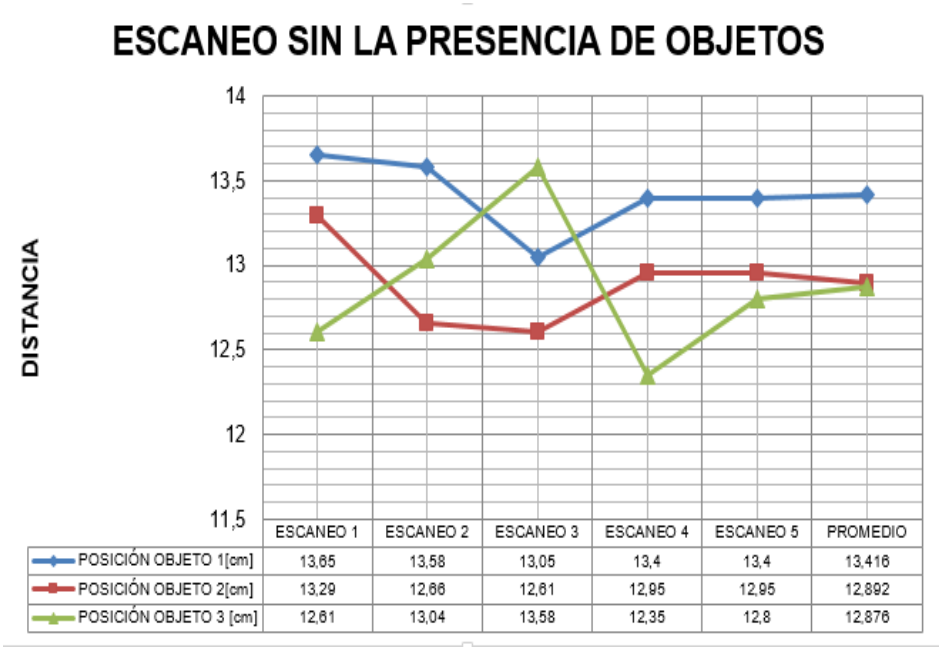


Figura 4.7 **Estadística de escaneo sin objetos**

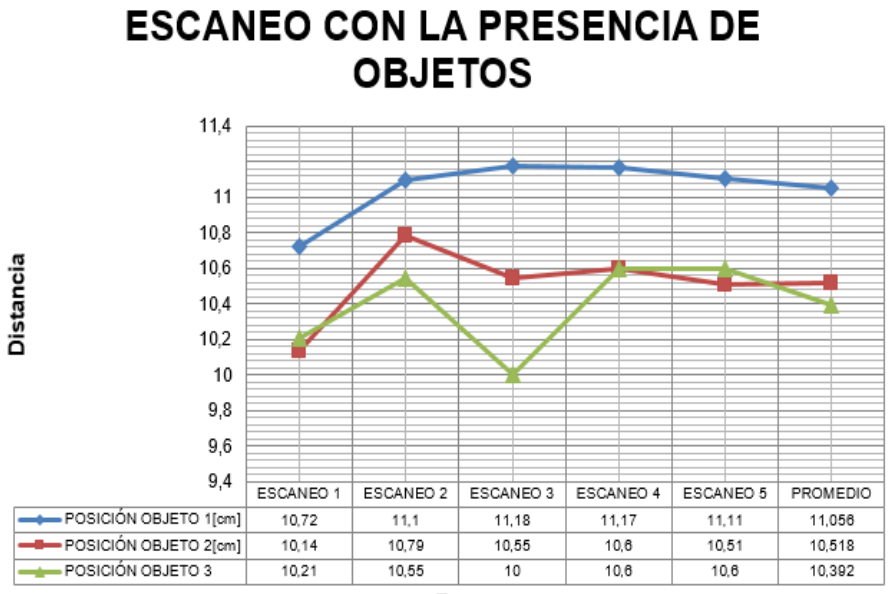


Figura 4.8 **Estadística de escaneo con objetos**

Una vez conocida el promedio de las distancias, encontramos el porcentaje de error para establecer la confiabilidad que nuestro proyecto tiene para decidir la presencia de un objeto.

DISTANCIA SIN OBJETO	VALOR TEÓRICO [cm]	VALOR EXPERIMENTAL [cm]	ERROR [%]
1	14,2	13,416	5,52
2	14,2	12,892	9,21
3	14,2	12,876	9,32

Tabla 4.3 Porcentaje de error de distancias sin objetos

DISTANCIA CON OBJETO	VALOR TEÓRICO [cm]	VALOR EXPERIMENTAL [cm]	ERROR [%]
1	11,7	11,06	5,50
2	11,7	10,52	10,10
3	11,7	10,39	11,17

Tabla 4.4 Porcentaje de error de distancias con objetos

4.3. ANÁLISIS DEL TIEMPO DE RECOLECCIÓN DE LOS OBJETOS

En la lógica del control del brazo robótico se establecieron retardos de tiempo, para que los desplazamientos de posición a posición no sean bruscos. Aun así se determina el tiempo que se demora en recolectar un objeto desde que la distancia es obtenida

por el sensor ultrasónico hasta que el objeto es dejado o abandonado en el punto de interés y se ubica en posición de escaneo para una nueva búsqueda.

PROCESO DE ESCANEO	TIEMPO RECOLECTAR OBJETO 1[s]	TIEMPO RECOLECTAR OBJETO 2[s]	TIEMPO RECOLECTAR OBJETO 3[s]
ESCANEO 1	12,17	12,63	13,71
ESCANEO 2	12,30	12,60	13,97
ESCANEO 3	12,11	12,58	13,43
PROMEDIO	12,19	12,60	13,70

Tabla 4.5 Tiempos de recolección de objetos

Como se puede apreciar, el menor tiempo obtenido es para el objeto más cercano y va aumentando conforme se va alejando del punto de recolección.

CONCLUSIONES

1. Con las pruebas realizadas del consumo de corriente de los servomotores, se aprecia que hay mayor consumo en las partes del Hombro, Codo y Muñeca del brazo robótico, razón por la cual en estas partes se usó servomotores de alto torque.

2. Las variaciones del ancho de pulso en la señal PWM que controla los servomotores provocan desplazamientos bruscos que dañan sus engranajes, para corregirlo usamos una subrutina que divide cada variación en múltiples incrementos o decrementos para suavizar el movimiento del brazo robótico.

3. Para percibir la presencia o no de un objeto establecimos un umbral de decisión en base a los datos obtenidos ya que en las pruebas de distancia realizadas con el sensor ultrasónico obtuvimos diferentes resultados tomando en cuenta la misma referencia.

4. La herramienta de desarrollo App Inventor, disminuye el tiempo de diseño de aplicaciones Android, ya que es una programación gráfica e intuitiva.

RECOMENDACIONES

1. Se recomienda trabajar sobre la computadora básica que Altera proporciona como ejemplo, pero adaptándola para nuestros requerimientos, con el afán de agilitar la creación de la misma.

2. Para que nuestra computadora básica sea portable, se recomienda integrar todos los componentes creados (PWM) a las librerías Qsys.

3. Para evitar errores de medición, se sugiere hacer el cálculo de la distancia contra un objeto, que se encuentre en un rango de 2cm a 400cm, según las especificaciones del sensor ultrasónico HC-SR04.

4. Para el adecuado funcionamiento del sensor ultrasónico, es decir para que pueda determinar la distancia correctamente, la fuente de alimentación de éste, debe proporcionar un voltaje entre (4.50 y 5.5 voltios) y una corriente de (2 y 15 mA).

5. Es recomendable acoplar la señal de control proveniente de la DE0-Nano hacia los servomotores, debido a que corrientes elevadas pueden quemar o dañar la tarjeta de desarrollo.

6. Se debe configurar el rango de operación de los servomotores individualmente ya que al aplicar la misma señal PWM se presentan diferentes posiciones angulares a las indicadas por el fabricante.

BIBLIOGRAFÍA

[1] AlteraAltera Corporation. (January de 1995 - 2014). *Embedded Design*. Obtenido de http://www.altera.com/education/univ/materials/boards/de0-nano/unv-de0-nano-board.html?GSA_pos=1&WT.oss_r=1&WT.oss=de0%20nano

[2] Definicion.de. (2008-2014). *Copyright © 2008-2014 - Definicion.de*. Obtenido de http://definicion.de/robotica/

[3] Federico Cristina, S. D.-F. (Agosto de 2012). *Android: Definiciones Básicas y Desarrollo de* . Obtenido de http://ftinetti.zxq.net/reptec/AndroidDocumentation-v1.pdf

[4] Jose Agraszal, L. P. (2010). *Robótica*. Obtenido de Historia de la Robótica: http://inteligencia-artificialrobotica.blogspot.com/p/historia-de-la-robotica.html

[5] Raúl Hernández Aquino, U. d. (9 de mayo de 2008). *Capitulo 1: Introducción al Bluetooh y wi-fi*. Obtenido de http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/hernandez_a_r/capitulo1.pdf

[6] Soria, K. (Martes de Septiembre de 2013). *Todo lo que tienes que saber sobre: HC-SR04 Sensor Ultrasónico*. Obtenido de http://bkargado.blogspot.com/2013/09/todosobrehc-sr04.html

[7] Towerpro. (2013). *TOWER PRO SERVO SPECIFICATION*. Obtenido de http://www.towerpro.com.tw/driver/drivers/Towerpro%20servo%20spec.pdf

[8] Wikipedia. (Enero de 2014). *Bluetooth*. Obtenido de http://es.wikipedia.org/wiki/Bluetooth

[9] Yamid Ramirez, U. P. (2010). *Servomotores*. Obtenido de http://www.monografias.com/trabajos60/servo-motores/servo-motores.shtml

ANEXOS

ANEXO A

MANUAL DEL SENSOR ULTRASÓNICO HC-SR04

Features

- Distance measurement range: 2cm - 400cm
- Accuracy: 0.3cm
- Detect angle: 15 degree
- Single +5V DC operation
- Current comsuption: 15mA

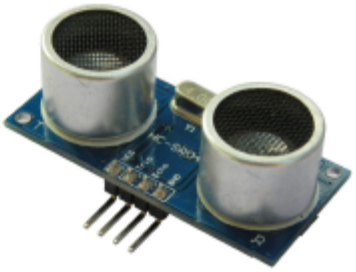


Fig. 1

How It Works

HC-SR04 consists of ultrasonic transmitter, receiver, and control circuits. When trigged it sends out a series of 40KHz ultrasonic pulses and receives echo from an object. The distance between the unit and the object is calculated by measuring the traveling time of sound and output it as the width of a TTL pulse.

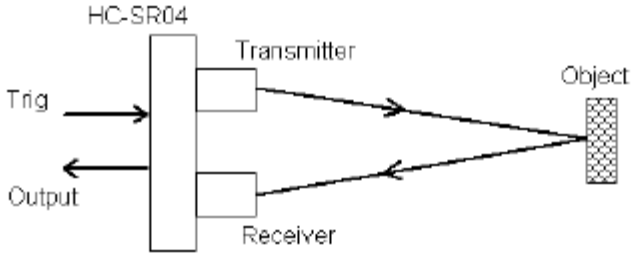


Fig. 2

How To Use It

To measure distance you need to generate a trig signal and drive it to the Trig Input pin. The trig signal leve must meet TTL level requirements (i.e. High level > 2.4V, low level < 0.8V) and its width must be greater than 10us. At the same time you need to monitor the Output pin by measuring the pulse width of output signal. The detected distance can be calculated by the formula below.

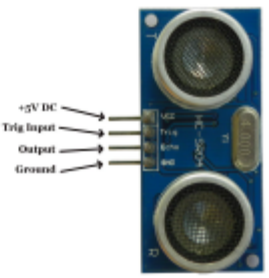


Fig.3

$$\text{Distance} = \frac{\text{Pulse Width} * \text{Sound Speed}}{2}$$

where the pulse width is in unit of second and sound speed is in unit of meter/second.

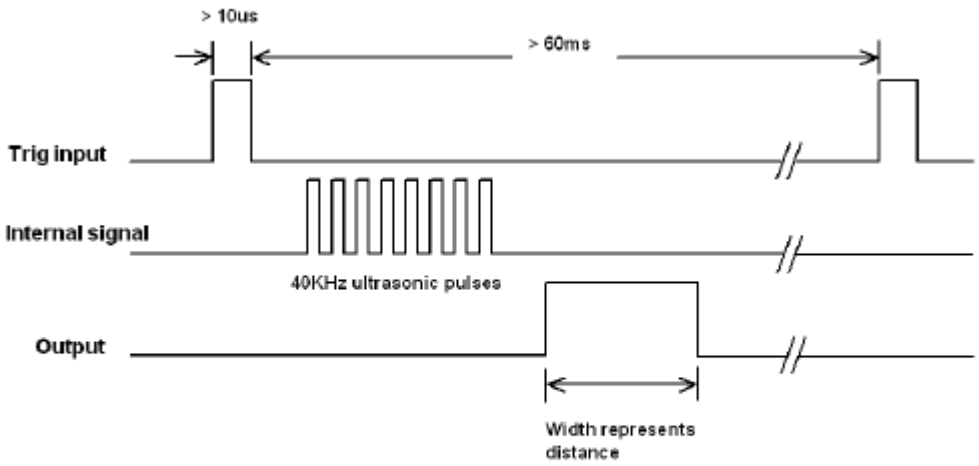


Fig. 4

Notes: 1. The width of trig signal must be greater than 10us
2. The repeat interval of trig signal should be greater than 60ms to avoid interference between connective measurements.

Specifications

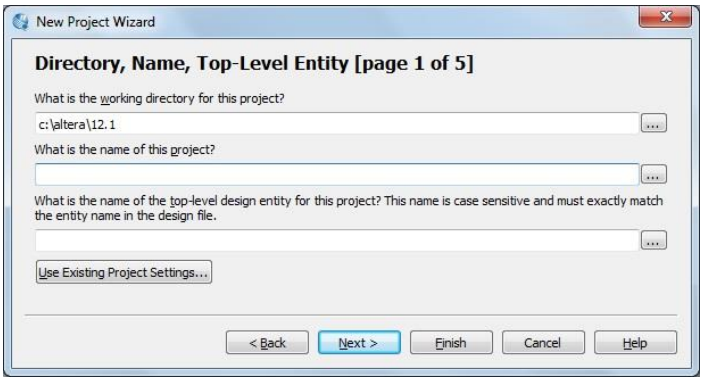
Parameters	Specification
Operating Voltage	+5V DC
Operating Current	15mA
Operating Frequency	40KHz
Maximum Distance	400cm
Minimum Distance	2cm
Detect Angle	15 degree
Resolution	0.3cm
Input Trig Signal	>10us TTL pulse
Output Signal	TTL pulse with width representing distance
Weight	
Dimension	45 x 20 x 15 mm

ANEXO B

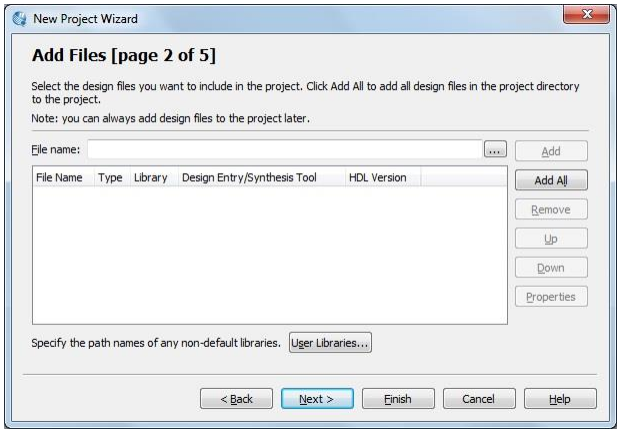
CREACIÓN DE UN NUEVO PROYECTO EN QUARTUS II

Una vez iniciado Quartus II, creamos un nuevo proyecto utilizando el asistente en el menú: File -> New Proyect Wizard…

En la ventana inicial debemos ingresar el nombre para el nuevo proyecto y el directorio de trabajo en donde se alojaran todos los archivos relacionados con dicho proyecto.



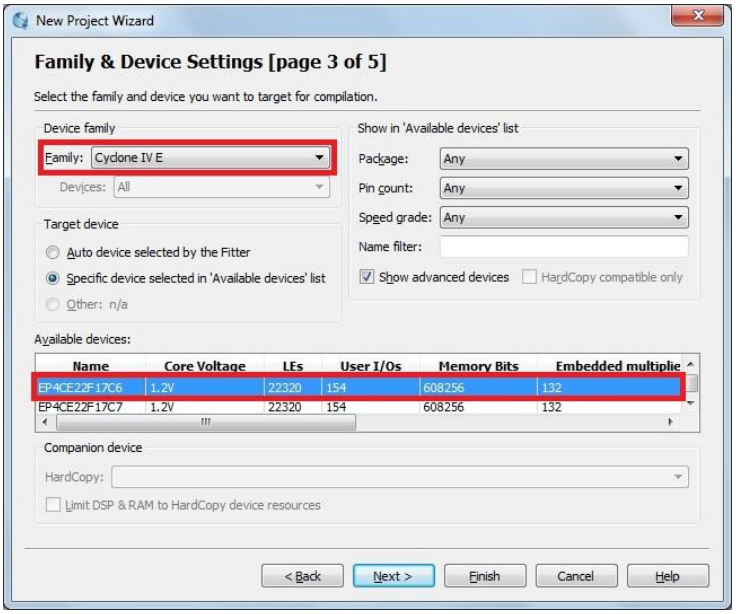
Como siguiente paso, se solicita agregar los archivos VHDL que necesitemos usar durante el desarrollo del proyecto (archivos existentes).



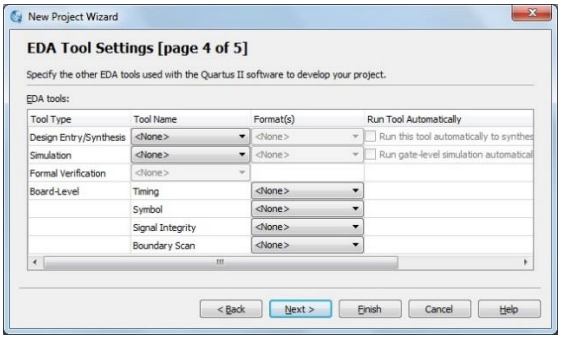
En el caso de tener archivos existentes, necesarios para la creación del nuevo proyecto y haberlos agregado, debemos indicar al programa en que dispositivo cargaremos el proyecto para su futura compilación. En este caso al tratarse de la tarjeta de desarrollo DE0-Nano debemos seleccionar los siguientes parámetros:

- En la sección Device Family.
  - Family: Cyclone IV E.
- En la sección Available devices.
  - EP4CE22F17C6

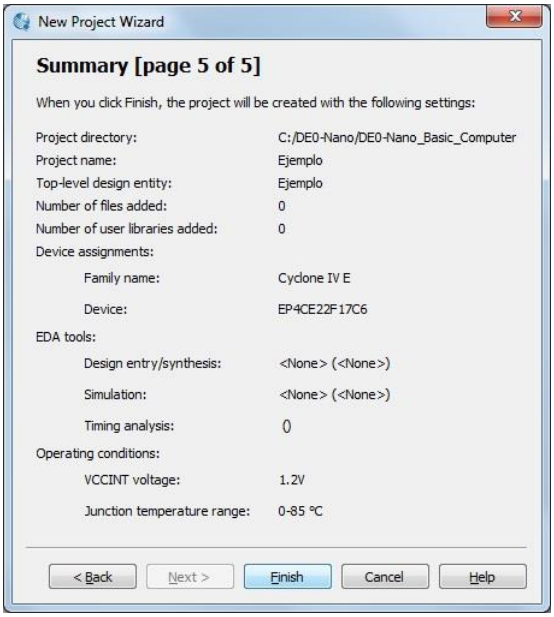
Una vez seleccionado el dispositivo que hace referencia a la tarjeta DE0-Nano tendremos una configuración similar a la de la Figura.



La siguiente ventana mostrada por el asistente, será en la cual debemos seleccionar las diferentes herramientas para la compilación y simulación del nuevo proyecto. Esta configuración se encuentra fuera del alcance de este documento ya que utilizaremos las herramientas estándares.



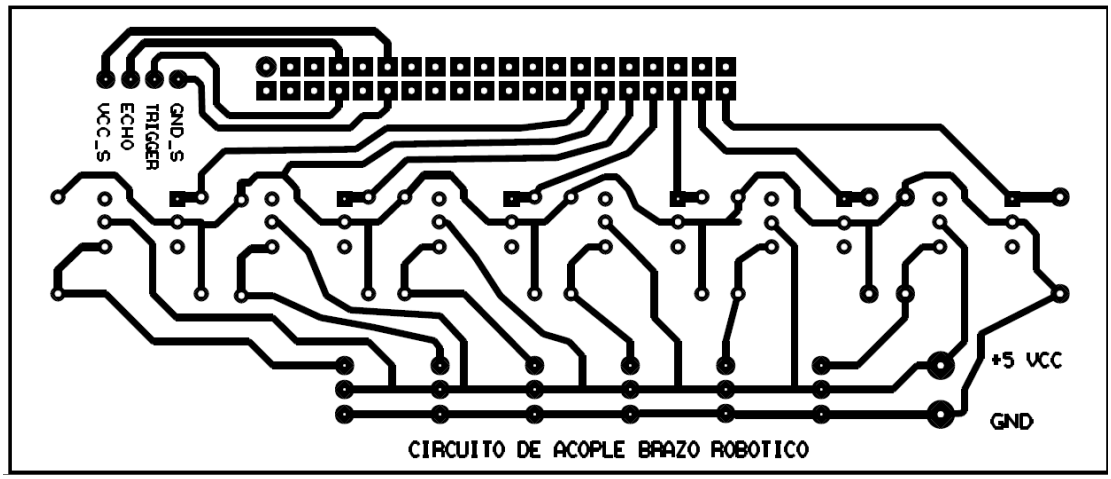
Finalmente el asistente nos muestra un resumen de toda la información y configuración realizada para el proyecto.



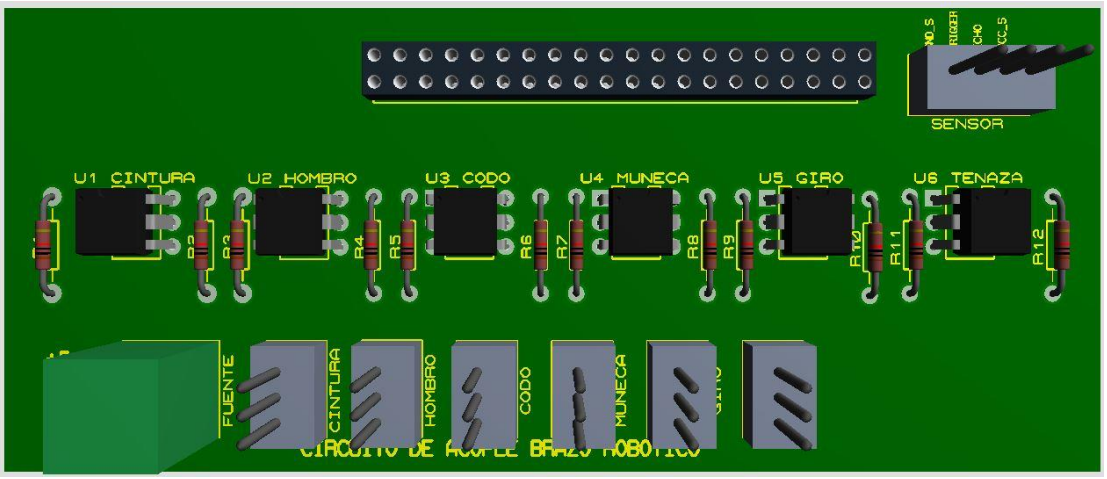
ANEXO C

DISEÑO DEL CIRCUITO DE ACOPLAMIENTO

Vista del PCB del cuircuito impreso



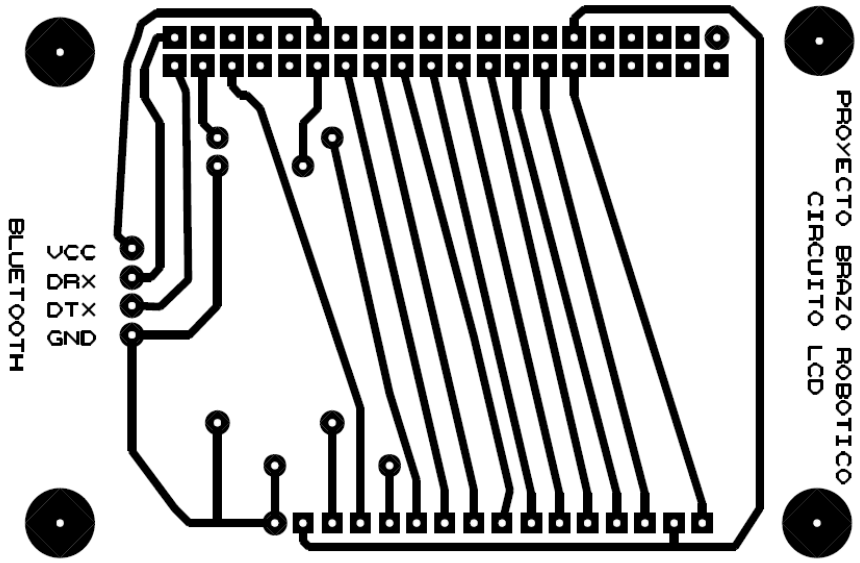
Vista del PCB de los componestes del circuito



ANEXO D

DISEÑO DEL CIRCUITO PARA LCD Y BLUETOOTH

Vista del PCB del cuircuito impreso



Vista del PCB de los componestes del circuito