ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

TESINA DE SEMINARIO

“CONTROL DE DIRECCIÓN PARA UN ROBOT CON ESTRUCTURA ACKERMAN STEERING, USANDO PLATAFORMA NIOS II**”**

Previa a la obtención del título de:
INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:
John Aníbal Rivera Burgos
José Javier Hernández Quinto

GUAYAQUIL – ECUADOR

2014

AGRADECIMIENTO

Agradezco profundamente a Dios, por guiarme en el sendero correcto de la vida.

A mis padres, por ser mi mayor fortaleza y mi guía, otorgándome la vocación de la lucha como heredad.

A mi pequeño hijo Ismael por brindarme la oportunidad de ser mejor cada día. A mis hermanos por apoyarme en cada decisión que tomo, y por estar a mi lado en cada momento hoy, mañana y siempre.

A mi director de tesis el Ingeniero Ronald Ponguillo por brindarnos sus conocimientos y consejos, permitiéndonos ser profesionales de calidad.

José Javier Hernández Quinto

AGRADECIMIENTO

A Dios, a mis Padres y hermana por su apoyo incondicional.

John Aníbal Rivera Burgos

DEDICATORIAS

Dedico este trabajo a mi pequeño Ismael, ya que él es el mayor de mis logros.

A mis padres Segundo Hernández y Victoria Quinto, que con su amor supieron enseñarme el valor de la verdad y la importancia de la familia.

José Javier Hernández Quinto

Dedico este trabajo a cada una de las personas que me apoyaron de manera incondicional.

John Aníbal Rivera Burgos

TRIBUNAL DE SUSTENTACIÓN

_____

Ing. Ronald Ponguillo I.

PROFESOR DEL SEMINARIO DE GRADUACIÓN

_____

Ing. Víctor Asanza

PROFESOR DELEGADO POR LA UNIDAD ACADÉMICA

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesina, nos corresponde Exclusivamente, y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

(Reglamento de Graduación de la ESPOL).

_____

Sr. José Javier Hernández Quinto

_____

Sr. John Aníbal Rivera Burgos

RESUMEN

El trabajo presentado en este proyecto, fue desarrollado para demostrar la potencialidad que contienen los microprocesadores embebidos en FPGA en la implementación de interfaces, permitiendo encaminar futuras investigaciones.

En el capítulo 1, Se describen los Objetivos Generales, específicos la identificación del problema y su metodología.

En el capítulo 2, damos a conocer información general de los conceptos que involucran el desarrollo de nuestro proyecto.

Se expone información detallada sobre las FPGAs, sistemas embebidos, la plataforma de desarrollo NIOS II de Altera, sensores de distancia, driver de motores y comunicación serial UART utilizado por el módulo Bluetooth.

En el capítulo 3, se presenta el diseño e implementación, además de la relación entre módulos, la funcionalidad y alcance acorde con los cálculos matemáticos del Sistema Ackerman.

El capítulo 4, se brinda la información correspondiente acerca del sistema operativo Android, utilizado en la creación de la interfaz de usuario, desarrollada en “software AppInventor” la cual se utiliza en la etapa de telecontrol del robot.

El capítulo 5, se muestran las pruebas realizadas, exponiendo tablas de datos en las cuales se analiza el comportamiento del sistema ackerman steering con la incorporación de la etapa de control, y fuerza.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

ABREVIATURAS

ADC	Conversor Analógico Digital
CPLD	Dispositivo lógico programable Complejo
E/S	Entrada y Salida
PCB	Circuito Impreso
FPGA	Arreglos de compuertas programables en campo
RS-232	Estándar de comunicaciones 232
GPIO	Propósitos Generales Input/Output
USB	Bus Universal Serial
JTAG	Grupo de acción conjunta de prueba
UART	Transmisor – Receptor Universal Asincronico
ASIC	Circuitos Integrados para aplicaciones especificas
EIA	Asociación de Electronica Industrial
PIC	Controlador de interfaz periférico

INTRODUCCIÓN

El crecimiento tecnológico, se podría denotar con las características que resalten de ciertos sistemas, como su capacidad de procesamiento, la versatilidad, excelente rendimiento, bajo costo. Hemos notado que el uso de procesadores embebidos basados en FPGA brinda una plataforma de desarrollo confiable, como son los dispositivos embebidos configurables.

Uno de los aspectos fundamentales de estos sistemas es la orientación académica que se brinda, posibilitando su exploración a gran escala, iniciando de esta manera nuevas tendencias de desarrollo, y la opción de acoplarlos con otros sistemas.

El desarrollo de nuestra temática se centra en controlar una estructura con mecanismo Ackerman Steering mediante un Sistema embebido, por su eficiente funcionalidad.

Además fue necesario elegir una plataforma de desarrollo adecuada para la interfaz entre el usuario y el prototipo desarrollado. Dada la reciente aparición del sistema operativo Android y su rápida adaptación en el mercado, acogido por numerosas marcas de terminales móviles y un alto número de usuarios, pareció una gran oportunidad para explorar su kit de

desarrollo y las posibilidades que ofrece. Además, fue un factor importante para tomar esta decisión el hecho de que sea un sistema operativo abierto, lo que permite al desarrollador llevar a cabo sus ideas y poder usarlas de primera mano en su dispositivo, así como ofrecerlas a la comunidad Android fácilmente.

CAPÍTULO 1

1 GENERALIDADES

Se inicia el documento identificando objetivos, problema y finalmente la metodología que se aplicar para el desarrollo del presente proyecto.

1.1 OBJETIVOS

1.1.1 OBJETIVO GENERAL

Se desea desarrollar un sistema embebido basado en el microprocesador NIOS II y bloques de lógica configurable que permita ejercer control sobre la dirección de un robot móvil terrestre con estructura Ackerman Steering.

1.1.2 OBJETIVOS ESPECÍFICOS

- Control de un prototipo robótico con estructura Ackerman Steering.
- Uso de componentes compatibles con un micro procesador embebido con propósitos específicos, desarrollado en Qsys, para la utilización

de ADC (Analog Digital Converter), PWM (Pulse Width Modulation) y comunicación serial UART.

- Desarrollo de un conjunto de algoritmos en NIOS II para la manipulación de sensores infrarrojos de distancia mediante el módulo ADC.
- Desarrollo de un conjunto de algoritmos en NIOS II para la manipulación de motores mediante el módulo PWM.
- Desarrollo de un conjunto de algoritmos en NIOS II para Comunicación Serial UART.
- Desarrollo de una aplicación basada en el Sistema Operativo Android utilizando el software AppInventor para el telecontrol de Robot vía bluetooth.
- Evaluar el funcionamiento del sistema mediante pruebas en ambientes controlados.

1.2 IDENTIFICACIÓN DEL PROBLEMA

Hoy en día existen prototipos robóticos que sin duda alguna son muy prometedores para la ayuda del hombre, actualmente en nuestro país el desarrollo de robots está en auge y es por la misma razón que hemos decidido colaborar con nuestros conocimientos para desarrollar el control de dirección de un robot con estructura Ackerman Steering, usando plataforma NIOS II.

Junto a una secuencia de investigaciones podremos observar su comportamiento, para poder dejar encaminado el desarrollo de este robot para futuras mejoras e investigaciones, incorporando nuevas y avanzadas tecnologías, para llevar a cabo su implementación final libre de errores y que sea de ayuda para el ser humano.

1.3 METODOLOGÍA

La elección del tipo de estructura es un aspecto importante para la construcción de un robot móvil, en nuestro caso se utilizó la estructura Ackerman Steering la misma que hace referencia a la forma en que se encuentran distribuidos los principales elementos que componen dicho sistema mecánico como son la plataforma, motores, ruedas, para lo cual adquirimos un modelo a escala de dicho sistema permitiéndonos utilizar la estructura sin ningún problema.

En la etapa de acoplamiento del móvil, se incluye una etapa de control, cuyos algoritmos serán diseñados en NIOS II.

La etapa de control está ligada a los periféricos que son controlados, entre ellos se denotan, los motores DC, Sensores Infrarrojos y dispositivo Bluetooth.

Se optó por un control asíncrono mediante Bluetooth, por su menor tasa de interferencia, además de permitir vincular o comunicar el robot con un dispositivo móvil. Por la acogida que tiene actualmente Android se ha decidido crear una aplicación con el fin de tele controlar al prototipo.

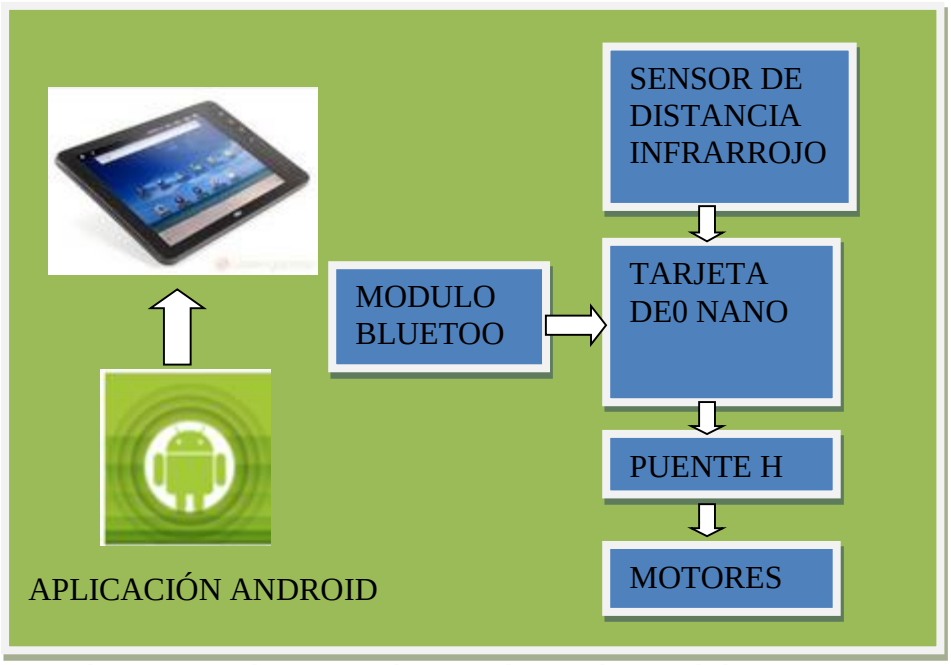


Figura 1.1 Diagrama de Funcionamiento del proyecto.

CAPÍTULO 2

2. MARCO TEÓRICO

En este capítulo se describe detalladamente las herramientas que se han utilizado para el desarrollo de nuestro proyecto.

2.1 SISTEMAS EMBEBIDOS

Se conoce como sistemas embebidos a una combinación de hardware y software de computadora, integrado en ocasiones a piezas electrónicas o de otro tipo, diseñado para tener una función específica. El uso de estos dispositivos es común, en diferentes aplicaciones. **[6]**

Esta combinación de software y hardware puede ser reemplazada en muchos casos por un circuito integrado que realice la misma tarea, pero una de las ventajas de los sistemas embebidos es su flexibilidad, ya que a la hora de realizar alguna modificación resulta mucho más sencillo modificar

pocas líneas de código al software del sistema embebido, que reemplazar todo el circuito integrado.

Las aplicaciones y algoritmos desarrollados deberán estar almacenados en memoria no volátil, como son las memorias ROM, flash y Compact Flash. **[4]**

Las principales características de un sistema embebido (ver figura 2.1) son el bajo costo y consumo de potencia. Muchos sistemas embebidos son concebidos para ser producidos en miles o millones de unidades, el costo por unidad es un aspecto importante a tener en cuenta en la etapa de diseño. Generalmente, los sistemas embebidos emplean procesadores muy básicos, relativamente lentos y memorias pequeñas para minimizar los costos.

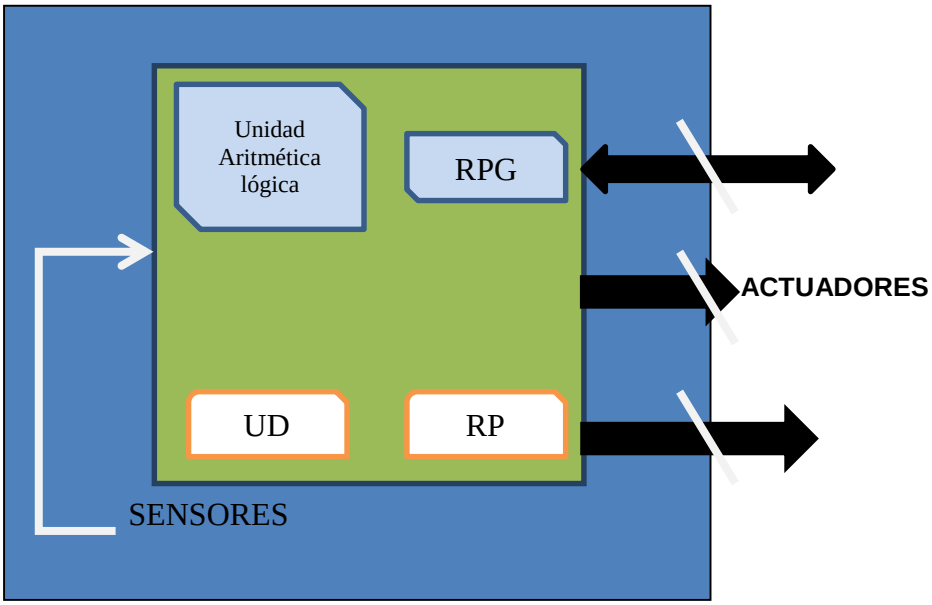


Figura 2.1 Esquema fundamental de un Sistema Embebido

Sus líneas de entrada/salida soportan la conexión de los sensores del dispositivo a controlar y todos los recursos complementarios disponibles, los cuales tienen como única finalidad atender a sus requerimientos.

2.2 NIOS II

El fabricante altera proporciona una infraestructura completa para la creación de sistemas en microprocesadores embebidos, permitiendo moldear las necesidades del diseñador, por medio de los componentes configurables que forman parte de las FPGAs. Véase figura 2.2

Por medio de la herramienta Qsys podemos configurar el microprocesador, que a través del programa Quatus II 12.1 se puede implementar directamente en la FPGA.

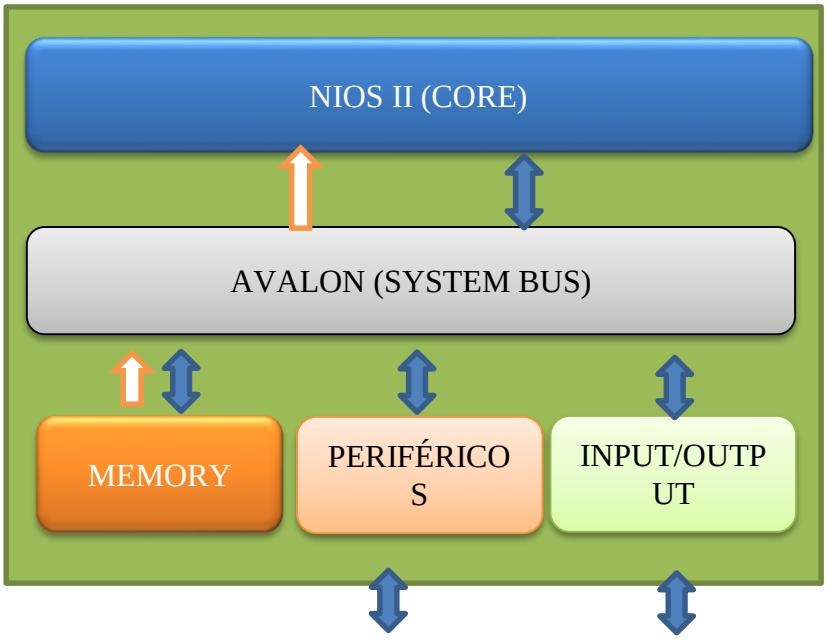


Figura 2.2 Infraestructura NIOS II de Altera

Como podemos observar en la figura 2.2, el sistema está compuesto por: El núcleo procesador NIOS II, memoria interna, periféricos integrados e interfaz para memoria externa y/o entrada salida. **[7]**

Al tratarse de un sistema flexible podemos adaptarlo a nuestras necesidades, obteniendo un alto rendimiento y disminución de costos.

2.2.1 VERSIONES

El NIOS II es un núcleo procesador configurable que se puede implementar en alguna de las tres versiones disponibles según se busque minimizar el consumo de recursos de la FPGA o maximizar el rendimiento del procesador:

- **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.

2.2.2 CARACTERÍSTICAS Y ARQUITECTURA

NIOS II es un procesador de 32 bits de propósito general, basado en una arquitectura tipo Harvard, dado que usa buses separados para instrucción, datos y cuyas principales características son:

- Tamaño de palabra asignado 32 bits.
- Juego de instrucciones RISC 32 bits.
- 32 Registros de propósitos generales.
- 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 memoria y periféricos externos.

2.2.3 ARQUITECTURA NIOS II

Como podemos observar en la figura 2.3 las principales unidades que encontramos en la arquitectura NIOS II:

- ✓ Los registros
- ✓ La unidad aritmética lógica
- ✓ La interface para instrucciones definidas por el usuario
- ✓ El controlador de excepciones

- ✓ El Bus de instrucciones
- ✓ El Bus de datos
- ✓ La memoria caché de instrucciones y datos
- ✓ La interface de acoplamiento directo de memoria de instrucciones y datos
- ✓ El módulo de depuración JTAG

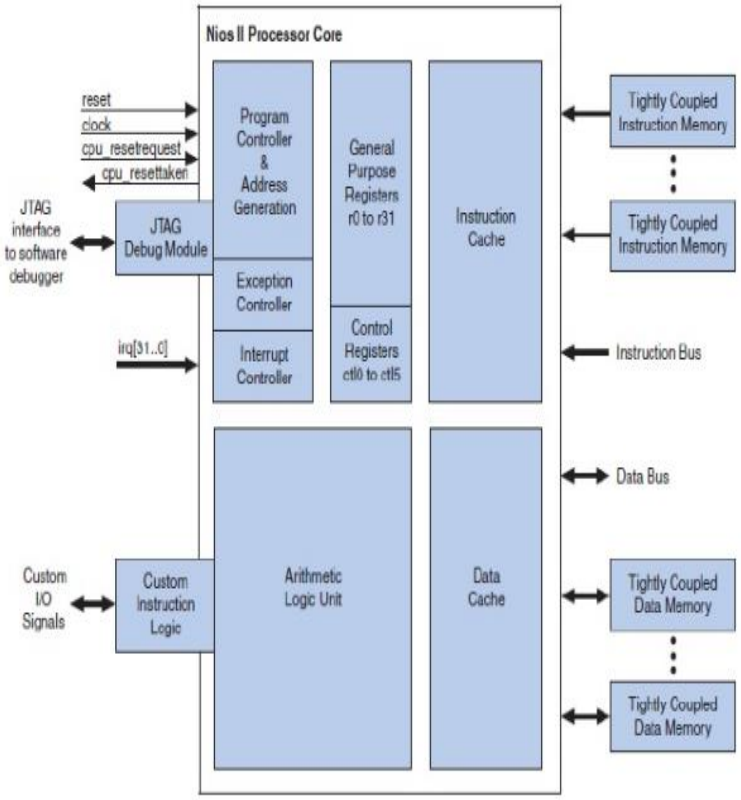


Figura 2.3 Diagrama de Arquitectura NIOS II

2.2.4 GESTIÓN DE EXCEPCIONES

El controlador de excepciones es el circuito encargado de realizar las tareas ligadas a la atención de excepciones, que son aquellas situaciones anormales que interrumpen el flujo de ejecución de un programa, dado que requieren la atención del procesador. **[8]**

2.2.4.1 EXCEPCIONES SOFTWARE

- **Instrucción trap ("software trap"):** se activa al transferir el control a un programa diferente, como un sistema operativo.

- **Instrucción no implementada ("unimplemented instruction"):** se produce cuando el procesador encuentra una instrucción válida que no ha sido implementada en hardware. Como es el caso de la instrucción de multiplicación y division en una implementación de II/e.

2.2.4.2 EXCEPCIONES HARDWARE

Interrupción hardware se produce cuando se da un evento sobre una de las 32 entradas de petición de interrupción de las que dispone el microprocesador (IRQ 0 a IRQ 31), lo que permite el acoplamiento asíncrono entre un periférico y el microprocesador.**[7]**

2.2.4.3 IDENTIFICACIÓN DE EXCEPCIONES

Cuando se produce una excepción, el procesador transfiere una ejecución al controlador de excepciones que las gestiona a través de una sola dirección. Esto provoca que las rutinas de atención deban determinar qué tipo de excepciones se han producido, agravando el tiempo de respuesta (latencia), por medio de la realización de las siguientes tareas:

Comprobar el registro de interrupciones pendientes (IPENDING) para verificar que se ha producido una interrupción hardware y poder lanzar la rutina de atención.

Comprobar que la instrucción Trap está siendo ejecutada al momento de la interrupción, siendo correspondiente a la dirección de retorno de excepciones EA menos 4 y poder lanzar la rutina apropiada.

2.2.4.4 CONTROLADOR DE INTERRUPCIONES

NIOS II incluye en todas las versiones un controlador interno de interrupciones (ICC). Además se puede añadir un bloque controlador de interrupciones externo (EIC).

El controlador de interrupciones internos gestiona las 32 entradas de peticiones de interrupciones (IRQ 0 a IRQ31) por medio del único vector de interrupciones.

El controlador de interrupciones externos proporciona un control de interrupciones vectorizado (VIC, Vector Interrupt Controller) con el que se da respuesta a la interrupción a través de vectores separados y según un determinado nivel de prioridad. Cuando se produce una interrupción se transfiere la ejecución directamente a la rutina de atención (ISR,Interrupt Service Rutine) apropiada, indicada por dicho vector a través de una instrucción de salto.

Cuando se produce una petición de interrupción (IRQ,Interrupt ReQuest), se realiza el siguiente proceso:

- Se cancela la instrucción en curso.
- Se resguarda información para reanudar el programa y valores de estados del CPU.
- Se deshabilitan las interrupciones externas al procesador.
- Se da el control al gestor de interrupciones para que identifique la fuente de interrupción y la marque en el registro de control de peticiones.
- Salta a la dirección donde se encuentra la rutina de atención de interrupción y borra la marca de petición.
- Se reanuda la ejecución del programa, una vez finalizada la rutina de atención de interrupción.
- Para que se genere una interrupción se debe cumplir lo siguiente:

- Que se encuentre activo el bit (PIE) del registro de estado que habilita las interrupciones de forma global.
- Que se produzca una llamada a una de las entradas de atención de interrupción IRQ.
- Que se encuentre activo el bit correspondiente a la entrada en el registro de control de interrupciones habilitadas (IENABLE).**[7]**

2.2.5 CONEXIÓN Y ACCESO A MEMORIA

El Microprocesador NIOS II utiliza 32 bits que son utilizados para el direccionamiento por byte, además de poseer buses diferentes, para instrucciones y datos.

En este tipo de conexiones el bus destinado a las instrucciones lee el programa ejecutado por el procesador, mientras que el bus destinado para los datos otorga acceso a los diferentes bloques de memoria y a los periféricos del sistema.

Existen tres maneras de acceder a memoria:

- Conexión a memoria a través de la red Avalon, tanto de memoria interna como externa.
- Conexión directa de bloques de memoria interna permite un rápido acceso a memoria.
- Conexión de memoria a través de caché (“caché memo”).

2.2.6 MEMORIA CACHÉ

El NIOS II puede incluir tanto caché de datos (NIOS II/f) como de instrucciones (NIOS II/f y NIOS II/s), implementadas en los diferentes bloques de memoria de la FPGA.

Al incluir memoria caché aumenta el rendimiento de la memoria principal, ubicada en un chip externo SDRAM.

La memoria caché al igual que la RAM se organiza en un arreglo tipo tabla, formado por filas y columnas ver figura 2.4.

En este caso el tamaño de caché es configurable. La caché de instrucciones se organiza en líneas de 8 bytes, mientras que la cache de datos se organiza en líneas de 4,16 y 32 bytes.

La gestión de caché se realiza por software, mediante la instrucción definida según sea caché de instrucciones o datos.

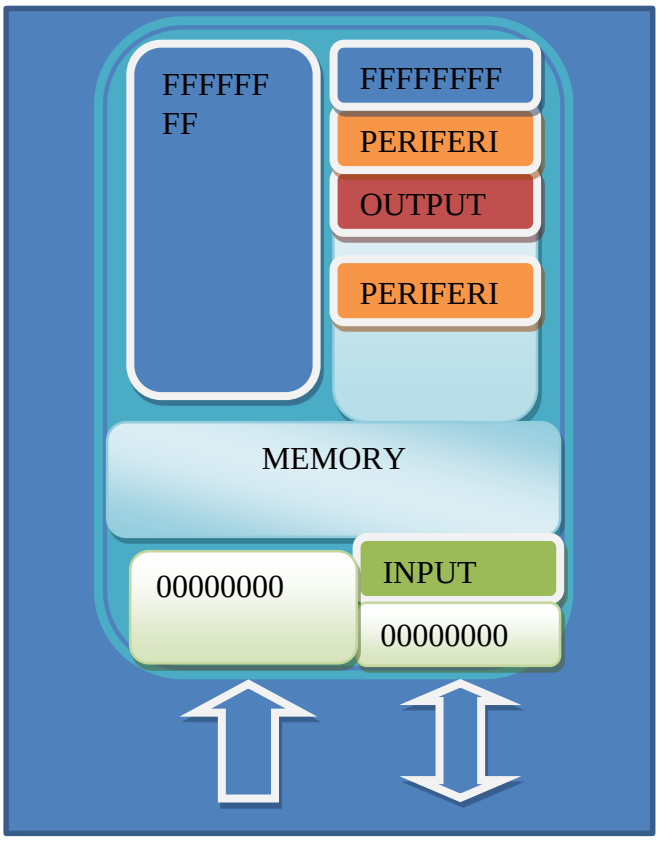


Figura 2.4. Organización de memorias y periféricos en NIOS II

Dado que el conjunto de periféricos y la capacidad de memoria se pueden configurar, el mapa de direcciones de memoria y periféricos (ver figura 2.4) es dependiente de la máquina, estableciéndose al momento de generar el sistema al igual que las direcciones de reset e interrupciones [7].

2.2.7 PERIFÉRICOS ESTÁNDAR

Altera proporciona un conjunto de periféricos comúnmente utilizados como los temporizadores, interface de comunicación serial, entrada/salida (I/O) de propósitos generales, controladores SDRAM y otras interfaces de memoria. **[8]**

2.2.7.1 PERIFÉRICOS A MEDIDA

La integración de periféricos a medida al sistema de NIOS II, puede ser utilizado a través de la interfaz de la red Avalon.

Además de permitirnos el añadir instrucciones propias a la unidad aritmética lógica (ALU), sobre todo pensando en aplicaciones de procesamiento digital de señal (DSP, Digital Signal Processing). **[7]**

2.2.8 RED AVALON

La Red Avalon nos permite la interconexión entre diferentes componentes, en este tipo de sistema hardware podemos identificar, cuatro diferentes conexiones que son:

- **Conexión en Bus:** se basa en la conexión de maestros y esclavos por medio de una unidad de arbitraje común, lo que permite operar a frecuencias relativamente altas a costa de perder la concurrencia.

- **Conexión Full CrossbarSwitch**: se basa en la conexión directa de maestro y esclavos por medio de un matriz de conexiones, lo que permite transacciones concurrentes entre los diferentes elementos del sistema.

- **Conexión PartialCrossbarSwitch**: se basa en la conexión directa entre fuente source y sumidero (sink) creando un flujo de datos unidireccionales para la transferencia de datos a alta velocidad , dado que se elimina la unidad de arbitraje al crear una conexión de punto a punto ,siendo uno de sus principales usos el procesamiento de videos.

  Altera basándose en una arquitectura "PartialcrossbarSwitch", implementa una serie de interfaces, una de ellas es Avalon que facilita la interconexión de componentes en sistemas complejos, y permite la interconexión de sistemas concurrentes, gracias a su arquitectura multimaestro.

2.2.8.1 INTERFACES AVALON

Existen seis diferentes tipos de interfaz:

- **La "AvalonMemoryMapped Interface" (Avalon-MM**) es una interfaz de lectura/escritura basada en direcciones, usado en conexiones maestro-esclavo.

- **La "AvalonStreaming Interface**" (Avalon ST) es una interfaz que soporta un flujo de datos unidireccional, incluyendo streams multiplexados, paquetes y datos DSP.

- **La "AvalonMemoryMapped Tristate Interface**" es una interfaz triestado de lectura y escritura basada en direcciones para el soporte de periféricos externos al chip. Así, múltiples periféricos puedes compartir buses de datos y direcciones para reducir el número de terminales de interconexión a la FPGA o el número de rutas en PBC.

- **La "AvalonClock"** es una interfaz que envía o recibe señales de reloj y reset para sincronizar interfaces.

- La "**AvalonInterrupt**" es una interfaz que permite que los componentes envíen señales de eventos a otros componentes.

- La "**AvalonConduit**" es una interfaz que proporciona señales para ser llevadas fuera del nivel del QSyS donde se puede conectar módulos adicionales diseñados sobre la FPGA.

De esta manera un componente puede incluir una o varias de estas interfaces, o varias instancias de la misma interfaz, otorgando acceso a la interacción entre sistemas complejos. **[8]**

2.2.9 GESTIÓN DE INTERRUPCIONES

En el NIOS II, el control de interrupciones se realiza por medio de los registros de control, los cuales requieren el uso de instrucciones de lectura y escritura especiales:

- La habilitación global de interrupciones se realiza por medio del primer bit del registro ctl0 (PIE).

- La desinhibición de cada una de las interrupciones (IRQ0 a IRQ31) se controla a través de los bits del registro ctl3 (IENABLE).

- Las interrupciones pendientes de atención se controlan a través de los bits del registro ctl4 (IPENDING).

La lectura y escritura de los registros son realizadas por medio de las instrucciones rdctl y wrctl.

- **RdctlrC, ctlN:** lee el valor del registro de control N y lo almacena en el registro rC.

- **WrctlctlN,rA:** escribe el valor del registro rA en el registro de control N.

2.3 ANDROID

2.3.1 DEFINICIÓN

Android es una plataforma de desarrollo para dispositivos móviles que permite el manejo de aplicaciones como:
Bluetooth, Wi-Fi, cámara, GPS, Acelerómetro, Infrarrojos, etc.

Posee un entorno de desarrollo muy elaborado mediante un SDK disponible de forma gratuita.

2.3.2 HERRAMIENTA DE DESARROLLO APPINVENTOR

El software que se utilizó tiene características propias para desarrollar en sistemas móviles suscritos por Android.

El Kit de desarrollo (SDK) nos permite diseñar los requerimientos de la aplicación como son los controles, adquisición de datos o segmentos de la programación orientados al enlace inalámbrico. Además se utilizó la herramienta de desarrollo (APP INVENTOR de Google), por la facilidad de manejo de algoritmos a través de bloques interactivos.

La aplicación o interfaz de usuario fue cargada en un dispositivo móvil, una Tablet que previamente fue configurada con las características del software requerido (Android Versión 2.3.1).

2.3.3 INTERFAZ GRÁFICA

Los dispositivos móviles han llegado a constituir una necesidad para aquellos que van más allá de las simples llamadas telefónicas o la ejecución de aplicaciones básicas. El gigante de Internet Google ha presentado un nuevo sistema operativo para este tipo de dispositivos Android.

Dada la reciente aparición del sistema operativo Android y su rápida expansión en el mercado, acogido por numerosas marcas de terminales móviles y un alto número de usuarios, pareció una gran oportunidad para explorar su kit de desarrollo y las posibilidades que ofrece (ver Figura 2.5).

Figura 2.5 Interface del desarrollador

La interfaz gráfica presenta el siguiente menú:

- Paleta de Herramientas
- Vista Principal
- Componentes
- Propiedades

- **Paleta de herramientas**

Nos permite arrastrar botones, cuadros de texto, slider, etc. Para ir formando la estructura deseada de nuestra aplicación en la presentación del menú de Vista Principal ver Figura 2.6.

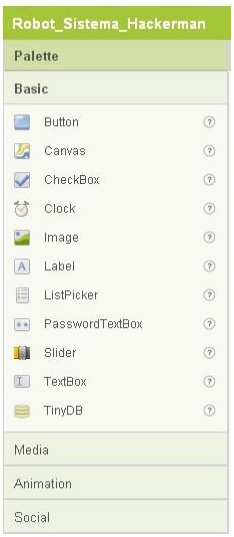


Figura 2.6 Vista Principal

En la Figura 2.7 se muestra la ventana inicial de AppInventor donde están los componentes necesarios para el desarrollo de la aplicación.

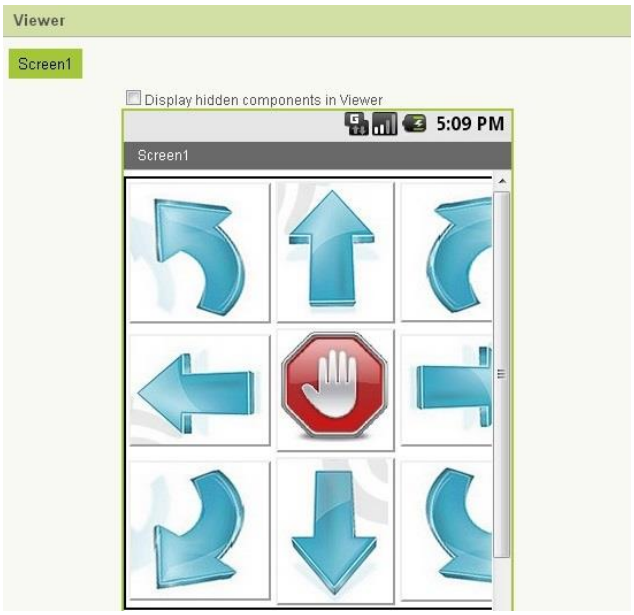


Figura 2.7 Ventana Vista de AppInventor

- **Componentes**

En la Figura 2.8 se muestran todos los componentes necesarios para la aplicación como es el BluetoothClient que nos permite hacer un enlace de datos con el accesorio Bluetooth que está conectado a la tarjeta De0Nano.

Figura 2.8 Ventana de Componentes de AppInventor

- **Propiedades**

Una vez diseñada la aplicación se procede a complementarla usando la ventana de propiedades (ver Figura 2.9.) para modificar tamaño de letra, colores, texto, alineación, etc.

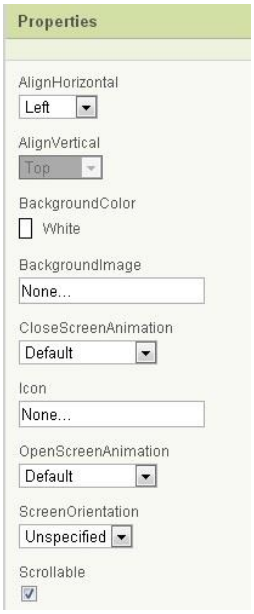


Figura 2.9 Ventana de Propiedades de AppInventor

2.3.4 DESARROLLO DE LA APLICACIÓN

En la Figura 2.10 se presenta la interfaz de funciones utilizadas para realizar el enlace vía Bluetooth y envío de comandos para el control del robot.

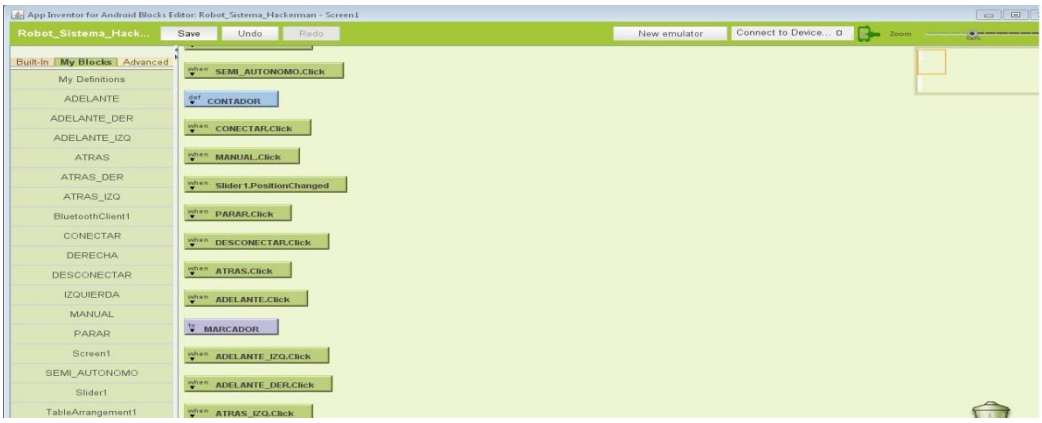


Figura 2.10 Ventana de comandos de AppInventor

2.4 TECNOLOGÍA BLUETOOTH

Es muy importante para establecer enlaces de comunicación inalámbrica (ver Figura 2.11) muchos dispositivos hoy en día utilizan esta tecnología, por ejemplo: Impresoras, teléfonos, consolas, teclados, altavoces, etc. Además es muy sencillo de activar, su alcance es limitado y en teléfono móvil puede ser vulnerable para la entrada de programas malignos.

Figura 2.11 Dispositivos con Tecnología Bluetooth

El alcance de esta tecnología depende de la clase a la que pertenezca, casi todos los dispositivos gobernados por este medio de comunicación pierde su conexión al incrementar su distancia. Si bien la Clase 2 llega sólo hasta los 10 metros, la Clase 1 alcanza hasta los 100 metros. Con ello, no obstante, se incrementa también el consumo de energía. Es por eso que la mayoría de los fabricantes se limita al alcance menor".

2.4.1 CARACTERÍSTICAS

- La tecnología inalámbrica Bluetooth está orientada al intercambio de señales como de voz y datos.
- Posee un ancho de banda de frecuencia de 2.4 GHz, que no requiere de licencia alguna.
- Tiene un radio de alcance de 10 o 100 metros dependiendo de la clase del dispositivo Bluetooth. La máxima velocidad de transmisión es de 3 Mbps.
- Los objetos sólidos no se manifiestan como obstáculo alguno para la tecnología inalámbrica Bluetooth.
- Tampoco se necesita que los dispositivos estén situados en la misma línea de visión es decir, orientados uno frente a otro, ya que se transmite en todas direcciones.

La tecnología Bluetooth permite que nuestro dispositivo móvil y el dispositivo HC06 (ver figura 2.12) se intercomuniquen y se comporten de forma discrepante: maestro y esclavo, de tal manera que el dispositivo inteligente móvil debe ser maestro y el robot es el esclavo. El dispositivo maestro, posee una aplicación que inicia la comunicación a través de sus pines de conexión denotadas en la tabla 2.1.

Figura 2.12 Módulo de comunicación Bluetooth HC06

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

Tabla 2.1 Pines Módulo Bluetooth

El dispositivo Bluetooth HC06 posee parámetros de configuración predeterminados de fábrica que se pueden observar en la siguiente tabla 2.2, que pueden ser reconfigurados por el usuario de acuerdo a sus requerimientos a través de comandos AT.

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

Tabla 2.2 Configuración inicial del dispositivo HC06

2.4.2 COMANDOS AT

Los comandos AT (Attention Command) son un conjunto de instrucciones que permiten relacionar al usuario con un dispositivo terminal Modem.

Los comandos AT son una cadena de caracteres ASCII que empieza con AT, después de ser procesados por el dispositivo devuelve la respuesta correspondiente a la petición requerida.

A continuación se presentan los comandos AT (ver tabla 2.3.) que nos permiten ingresar al procesador del dispositivo Bluetooth y verificar parámetros para que la comunicación se establezca de forma exitosa.

CONFIGURACION HC06 POR COMANDOS AT		
PETICION	**COMANDO**	**RESPUESTA**
Test de comunicación	AT	OK
Baud Rate a 9600	AT+BAUD4	OK9600
Paridad Ninguna	AT+PN	OK NONE
Paridad par	AT+PE	OK EVEN
Paridad Impar	AT+PO	OK ODD
Versión	AT+VERSION	OKlinvorV1.5
Nombre	AT+ESPOBOT	OKESPOBOT

Tabla 2.3 Configuración HC06 por comandos AT

Para la configuración de los parámetros del módulo Bluetooth se utilizó AccessPort, que es un software de comunicación serial.

En la Figura 2.13 se muestra la configuración a 9600 baudios por medio del software AccessPort a través de comando AT.

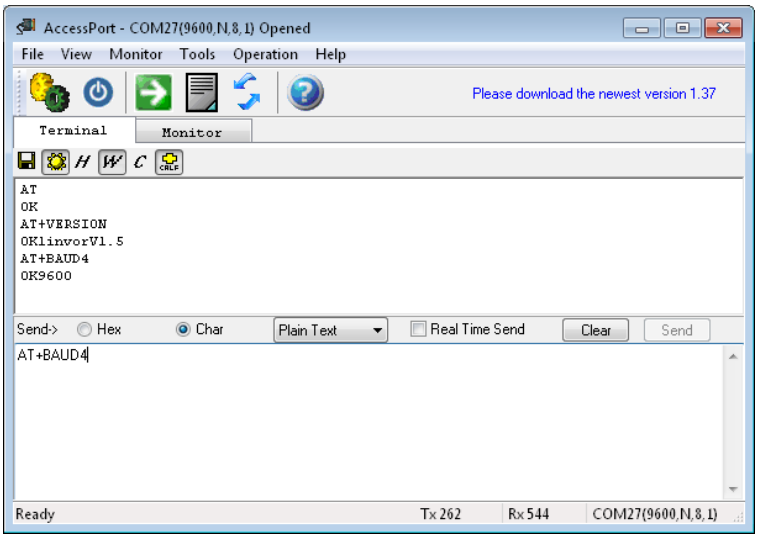


Figura 2.13 Configuración del Baud Rate del Módulo Bluetooth HC06

2.5 COMUNICACIÓN SERIAL ASÍNCRONA

Es un método de comunicación por el cual se transmite y se recibe un caracter, añadiendo bits de inicio y al final del paquete de datos transmitido, con esto se sincroniza el transmisor y receptor con el fin de separar los caracteres.

Se definió el modo UART como una clase de comunicación serial asíncrona como se observa en la figura 2.14

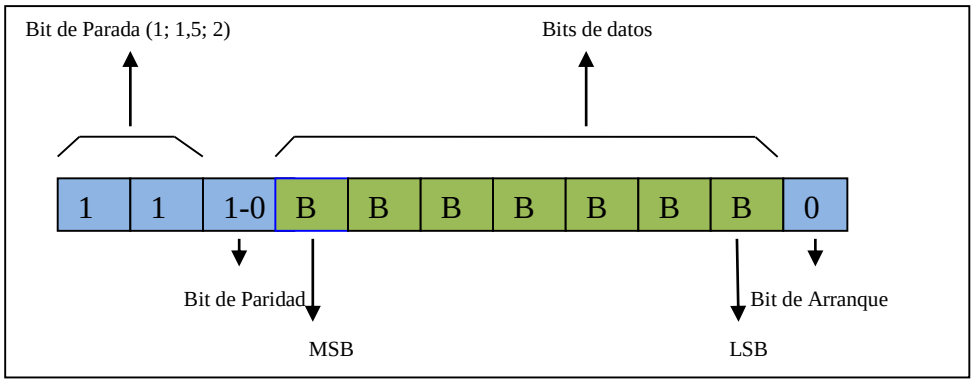


Figura 2.14 Formato de datos asíncronos

2.5.1 MODOS DE INTERCAMBIOS DE DATOS

Conforme los datos tienen su formato para el intercambio entre el transmisor y receptor, su comunicación puede clasificarse como.

- Simplex (SX).
- Half-Duplex (HDS).
- Full-duplex (FDX).
- Full/Full-Duplex (F/FDX).

Modo Simplex

Es el modo en el cual un sistema de comunicación se produce en un solo sentido (ver figura 2.15). En el sistema solo el maestro podrá transmitir mas no recibir, mientras el esclavo solo podrá recibir.

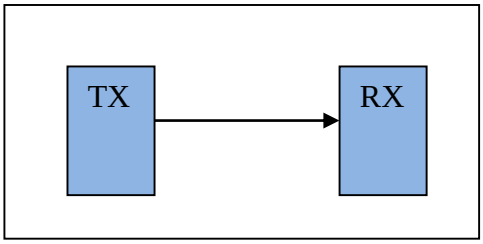


Figura 2.15 Modo Simplex

Modo Half Duplex

En este método ambos puntos pueden transmitir los datos pero, uno a la vez (ver figura 2.16).

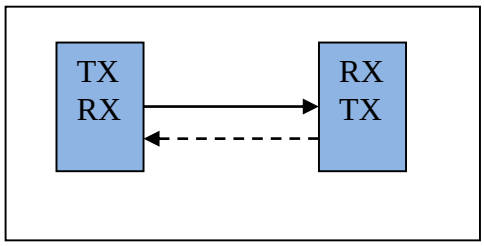


Figura 2.16 Modo Half Duplex

Modo Full Duplex

En este medio ambos pueden transmitir su información al mismo tiempo (ver figura 2.17).

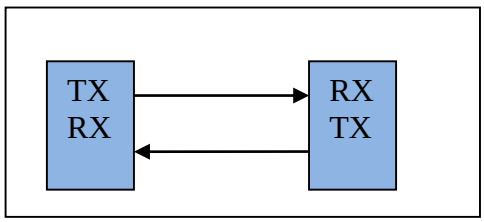


Figura 2.17 Modo Full Duplex

Modo Full/Full Duplex

En este método de operación, se puede transmitir y recibir al mismo tiempo y desde varios puntos a la vez (ver figura 2.18).

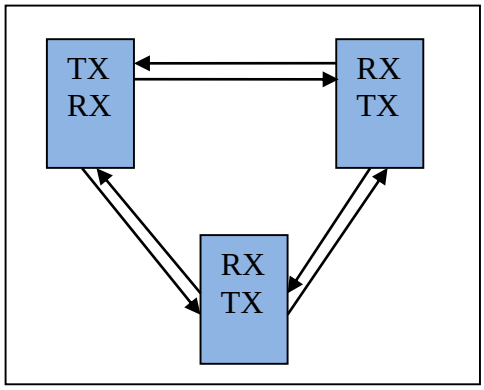


Figura 2.18. Modo Full/Full Duplex

2.5.2 COMUNICACIÓN SERIAL UART

Es un modo de comunicación que se puede transmitir y recibir caracteres de forma asíncrona desde un dispositivo a otro, este enlace se lleva a efecto por la misma razón de baudios configurados previamente, por lo tanto utiliza una línea de comunicación full dúplex(FDX).

2.5.3 CARACTERÍSTICAS DEL PROTOCOLO UART

Este protocolo presenta características descritas a continuación:

- Configuración de paridad par, impar y sin paridad para datos de 7 u 8 bits.
- El bit menos significativo es el primero en ser transmitido y recibido.
- Posee buffer de transmisión y recepción separado.

- Configuración de tasa de bits por segundo (Baud Rate) desde (110 hasta 256000 baudios).
- Capacidad para levantar interrupciones por transmisión y recepción de dato.
- Tiene bit de parada (1; 1,5; 2).

2.6. DISPOSITIVO HC06

Es un módulo con tecnología Bluetooth que sirve para el intercambio de datos de forma inalámbrica. En la tabla 2.4 se muestra algunas características del dispositivo.

CARACTERISTICAS HC06	
Protocolos soportados	Serial Uart, USB, SPI
Frecuencia de trabajo	2.4 GHz
Rango de trabajo	3.1V~4.2V
Corriente en comunicación	8 mA
Bluetooth Clase	1
Alcance máximo	100 metros
Temperatura soportada	-25°c hasta + 75°c

Tabla 2.4 Características del dispositivo Bluetooth HC06

2.7 ADC (CONVERSIÓN ANALÓGICA DIGITAL)

Los tipos de parámetros en que se puede representar una señal son: Analógicas y digitales**.**

Las señales analógicas se caracterizan por ser continuas en el tiempo, a lo largo de un rango determinado. Por ejemplo: Una señal sinusoidal (ver figura 2.19).

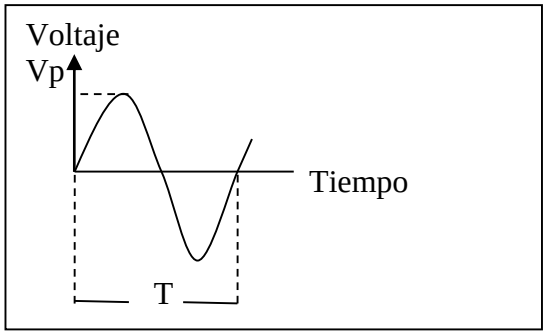


Figura 2.19 Representación de una señal analógica

Las señales digitales están determinadas por intervalos discretos de tiempo y valores discretos como se muestra en la Figura 2.20

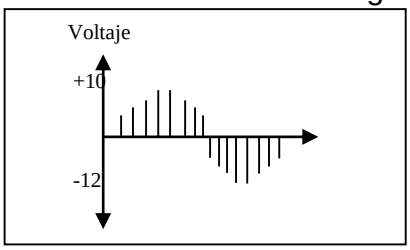


Figura 2.20 Señal cuantificada, codificada y cuantizada.

2.7.1 PROCESO DE DIGITALIZACIÓN DE UNA SEÑAL

En un proceso de digitalización de una señal analógica se muestrea normalmente a una razón constante. La señal muestreada se codifica para obtener un número binario.

Esta señal muestreada se mantiene constante durante su proceso de codificación ya que se necesita de un tiempo finito no igual a cero.

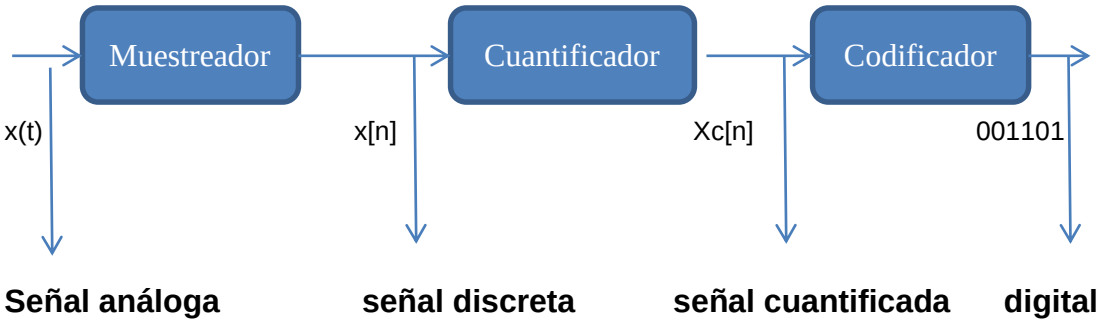


Figura 2.21 Convertidor A/D

2.8 SENSOR INFRARROJO

Esta tecnología se inició en los años 90, poseen un transductor que les permite convertir la radiación emitida por los materiales calientes en una señal eléctrica. Poseen un transmisor y un receptor que requieren una comunicación lineal.

2.8.1 CARACTERÍSTICAS

Los sensores están sellados con resina epóxica para ser inmunes al medio natural, su rango de funcionamiento máximo es de 5 metros.

El dispositivo emite un rayo infrarrojo codificado para ser inmune a la luz ambiente y es infrarrojo para ser invisible para el ser humano.

Se comunican por ondas de muy alta frecuencia (ver figura 2.22), además de algunas limitaciones como el ángulo y distancia. [10]

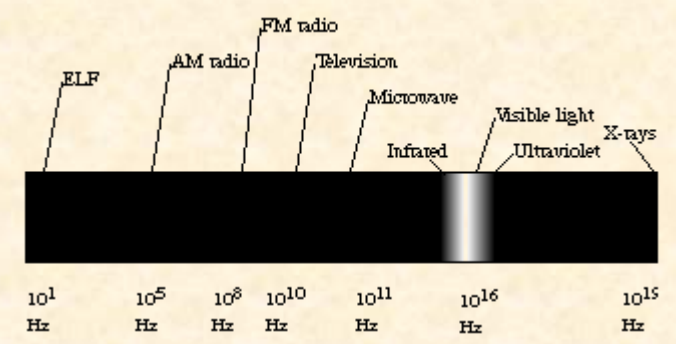


Figura 2.22 Frecuencia de Onda Infrarroja

2.8.2 LIMITACIONES

Su velocidad de transmisión de datos es muy lenta, por ejemplo: Si se requiere de transferir un archivo de 4Mb puede tardar entre 15 y 20 minutos.

2.8.3 VENTAJAS

- Requerimientos de bajo voltaje por lo tanto es ideal para Laptops, teléfonos, asistentes personales digitales.
- Sensor de bajo costo.
- Circuitería simple: no requiere hardware especial, puede ser incorporado en el circuito integrado de un producto.
- Alta seguridad: Como los dispositivos deben ser apuntados casi directamente alineados (capaces de verse mutuamente) para comunicarse.

2.8.4 DESVENTAJAS

- Se bloquea la transmisión con materiales comunes: personas, paredes, plantas, etc.
- Corto alcance
- Sensible a la luz y el clima. Luz directa del sol, lluvia, niebla, polvo pueden afectar la transmisión.
- Velocidad: la transmisión de datos es más baja que la típica transmisión cableada.

2.9 DISPOSITIVO GP2D12

Es un dispositivo medidor de distancia que procesa señales infrarrojas mediante un transductor que permite obtener salida de voltaje analógico (ver figura 2.23).

Figura 2.23 Sensor infrarrojo GP2D12

Se muestra en la Figura 2.24 el diagrama de bloques del sensor GP2D12.

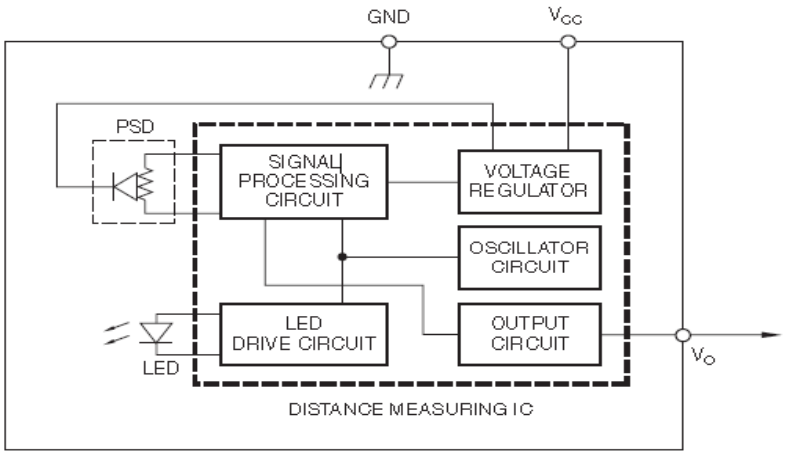


Figura 2.24 Diagrama de bloques GP2D12

2.9.1 CARACTERÍSTICAS DEL SENSOR GP2D12

- Rango de medición de distancia (10cm – 80 cm).
- Corriente máxima de 33mA con distancia máxima de 80 cm.
- Voltaje de alimentación (4.5 – 5 voltios).

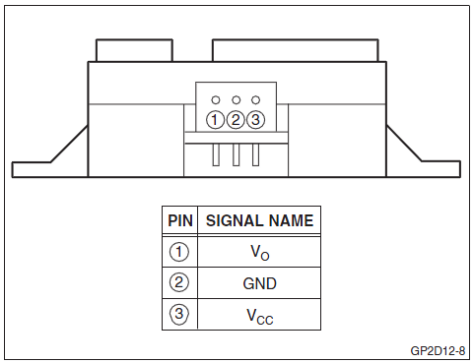


Figura 2.25 Pines de salida GP2D12

La gráfica representa la curva del voltaje de salida analógico Vo con respecto a la distancia medida en centímetros mostrada en la figura 2.26).

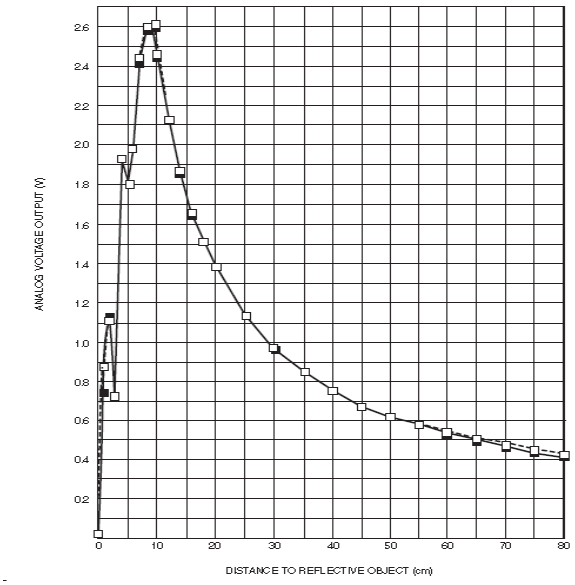


Figura 2.26. Gráfico Vo [v] vs Distancia [cm]

Se ha determinado una ecuación que permite obtener una línea grafica dada por su fabricante.

Esta fórmula (Ver ecuación 1) es clave para la lectura de distancia.

$$Vout = \frac{1}{\mathrm{L}+0.42}\ [\frac{1}{\mathrm{cm}}] \qquad \textbf{Ecuación (2.1)}$$

Parámetros:

Vout = Voltaje de salida [voltios].

L = Distancia [metros]

Se muestra una curva suavizada el cual facilita su entendimiento entre la relación de voltaje y su inverso de distancia (ver figura 2.27).

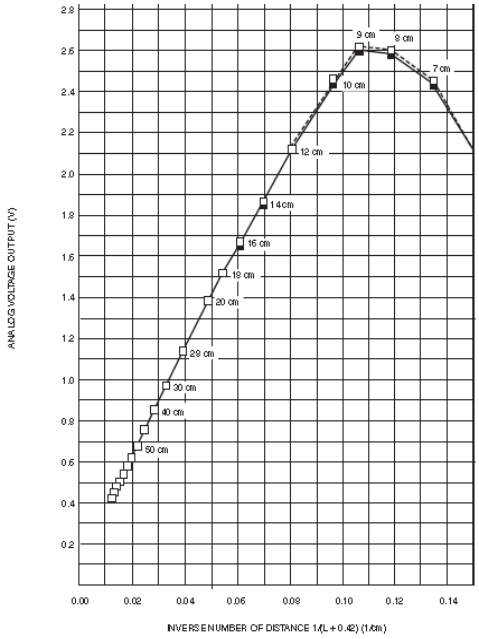


Figura 2.27 Voltaje vs Inverso de Distancia

2.10 PWM (MODULACIÓN POR ANCHO DE PULSO)

PWM (Pulse Width Modulation), es un sistema el cual consiste en la generación de una onda cuadrada en la que se varia el tiempo para que su valor activo en alto (ver figura 2.28) se conserve, manteniendo el mismo periodo.

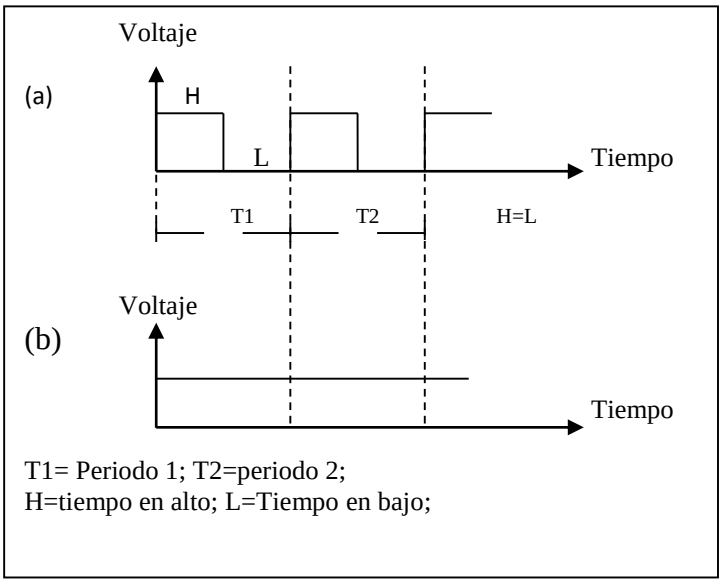


Figura 2.28 PWM Porcentajes de comparación 50%(a) 100%(b)

2.10.1 DRIVER PUENTE H

Es un circuito electrónico que permite girar a un motor de corriente continua en los dos sentidos, alimentado por una misma fuente de voltaje.

Existen muchos dispositivos puente H que están fabricados como circuitos integrados, aunque también se los puede construir por componentes discretos.

Se presenta un esquema básico de un Puente H en la figura 2.29.

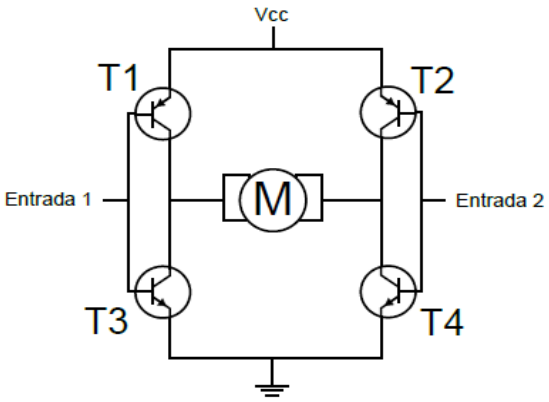


Figura 2.29 Esquema del Puente H

2.10.2 DISPOSITIVO LM298

Se diseñó un Puente H en base al circuito L298 (Ver figura 2.30**)** con la capacidad de controlar dos motores de corriente directa simultáneamente, se cuenta con un bloque de alimentación (Voltaje de alimentación, Voltaje de control y tierra), dos bloques de señales de control y dos bloques para la conexión de los motores.

El voltaje control es una referencia que utiliza el circuito integrado, es necesario conectarlo directamente al voltaje del dispositivo que emite las señales de control.

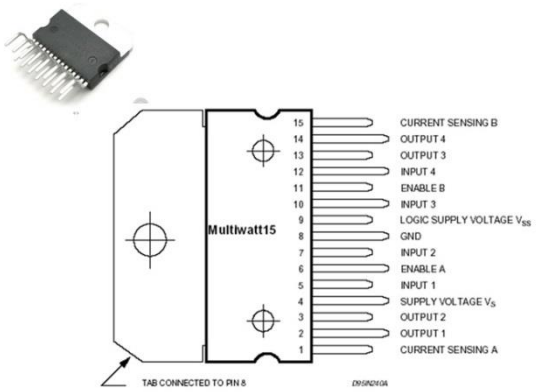


Figura 2.30 Circuito Integrado LM298

Características y Funciones

- Amperaje máximo 2A.
- Voltaje de operación de 4.5V a 46V.
- Señales de control de 4.5 a 7V.
- Control bidireccional del giro.
- Control de velocidad (Mediante técnica PWM).
- Control de dos motores simultáneamente.

2.11 MOTORES DC

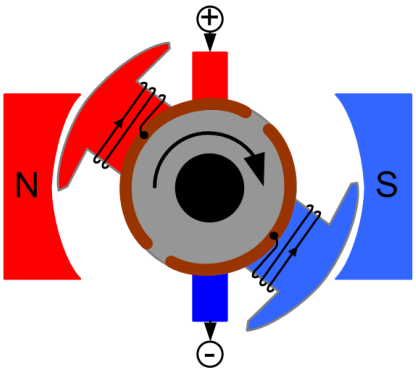


Figura 2.31 Motores DC

En figura 2.31 observamos un motor DC (Corriente Directa) o también llamados CC (corriente continua) de los más utilizados en el ambiente robótica varían por tamaño y potencia, pero todos se basan en el mismo principio de funcionamiento.

Su funcionalidad es básicamente aplicar fuente de alimentación entre sus bornes, para invertir el sentido de giro basta con invertir la polaridad.

Los motores DC no pueden ser posicionados o enclavados en una posición a diferencia de los motores pasos a paso o los servomecanismos.

CAPÍTULO 3

3. MECANISMO ROBÓTICO

En este capítulo se presenta información sobre con los robots móviles y los parámetros utilizados en el diseño del robot.

Además se describe las diferentes configuraciones con múltiples grados de libertad, centros de rotación y los tipos de rueda.

3.1 ROBOTS MÓVILES CON RUEDAS

La evolución de la robótica y los avances tecnológicos permiten diseñar robots de uso didáctico implementados en colegios y universidades. **[10]**

Los robots se clasifican de la siguiente manera:

- Humanoide.
- Robot móvil.

- Robot industrial.
- Robot inteligente.
- Robot de servicios.

En este trabajo se desarrolló un robot móvil con ruedas con características Ackerman, por el direccionamiento de las llantas delanteras conectadas al volante.

3.2 ESTRUCTURA GENERAL DE UN ROBOT MÓVIL

El diseño de un robot móvil es basado en el comportamiento del ser humano, en la Figura 3.1 se muestra la semejanza de la estructura de los seres vivos y un robot.

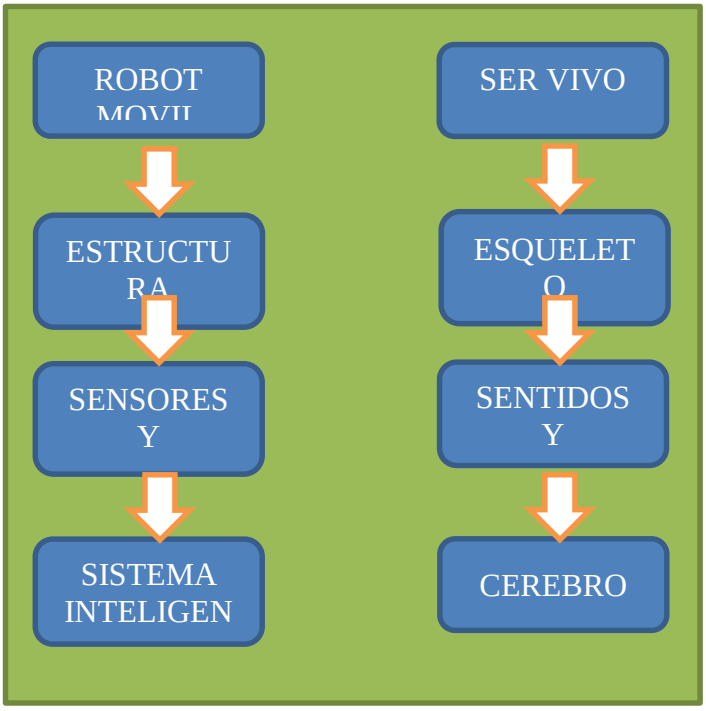


Figura 3.1 Comparación entre sistemas vivientes y robóticos

La estructura de un robot móvil presentada en la figura 3.1, está formada por diferentes subestructuras, tales como:

Estructura mecánica: estructuras con ruedas, patas y orugas.

Actuadores: motores, luces, brazos, ruedas y otros elementos que permita interactuar con el entorno.

Sensores: sonar, infrarrojos, ultrasónicos, laser, cámaras y cualquier elemento que nos proporcione información del entorno.

Inteligencia: métodos, algoritmos, etc. Estos permiten a través de la información de los sensores, tomar decisiones e interactuar con el entorno.

3.3 GRADOS DE LIBERTAD Y TIPOS DE RUEDAS

Los grados de libertad de un robot, así como los tipos de ruedas son aspectos que intervienen en el proceso de control y análisis de movimiento del robot.

3.3.1 GRADO DE LIBERTAD (GDL)

Es cada uno de los movimientos de desplazamiento y rotación que puede realizar el robot.

- Un cuerpo que se mueve en dos dimensiones tiene 3 GDL (una rotación y 2 traslaciones).
- Un cuerpo que se mueve en tres dimensiones tiene 6 GDL (3 rotaciones y 3 traslaciones).

3.3.2 SISTEMA HOLONÓMICO Y NO HOLONÓMICO

Un sistema es holonómico si la cantidad de grados de libertad que se pueden controlar es igual a la cantidad de grados de libertad disponibles. En un sistema no holonómico el robot móvil no podrá desplazarse lateralmente. En un sistema no holonómico, no es suficiente conocer la distancia recorrida para calcular la posición final de robot. Sino conocer como fue ejecutado el movimiento, en función del tiempo.

3.3.3 TIPOS DE RUEDAS.

Las ruedas son unos de los elementos importantes que permite la movilidad en un robot y se clasifican como:

- **Rueda fija.** Figura 3.2 (a). El movimiento se produce en la dirección de la rueda. Dónde:
  - ω.- es la velocidad angular de la rueda.
  - V.- la velocidad.
  - Ax.- el vector unitario de dirección.

- **Orientación centrada.** Figura 3.2 (b). Además del giro t de la rueda, existe rotación alrededor del eje vertical que está dirigido al centro de la rueda.

- **Orientación descentrada.** Figura 3.2 (c). Gira sobre el eje de la rueda y rota alrededor del eje vertical situado a una distancia d desde el centro de la rueda.
- **Rueda sueca.** Figura 3.2 (d). Permite moverse en la dirección de la rueda y perpendicular a ella.

$$V=(r*\omega)ax+Uas.$$

Ecuación 3.1

Dónde:

U.- es la velocidad de deslizamiento.

As.- es un vector unitario en la dirección del deslizamiento es la velocidad angular de la rueda.

V.- es la velocidad.

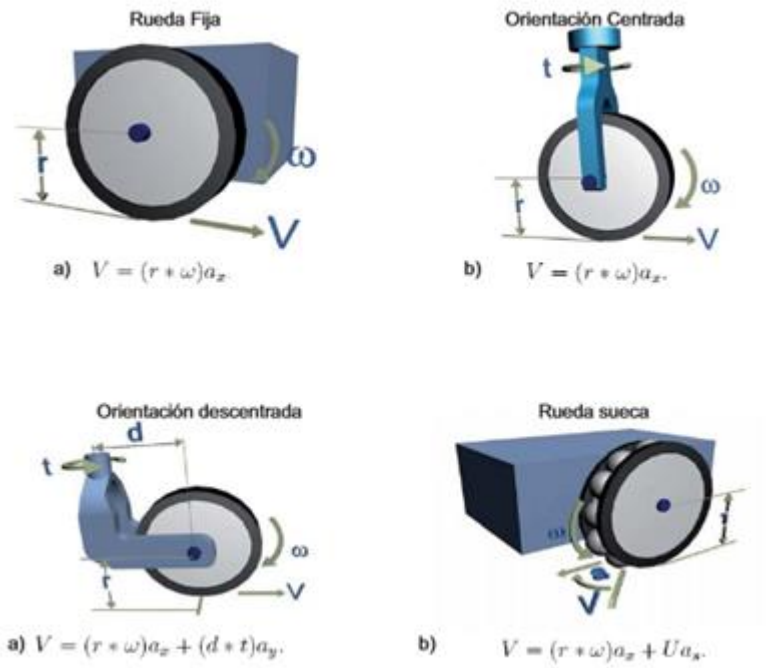


Figura 3.2 Tipos de Rueda

3.3.4 CENTRO INSTANTÁNEO DE ROTACIÓN.

Se define como el punto por el cual cruzan los ejes de todas las ruedas permitiendo que el robot gire. En la figura 3.3 se muestra el ICC para la configuración diferencial, Ackerman y triciclo.

La elección del tipo de configuración es uno de los pasos necesarios para la construcción de un robot móvil, la configuración hace referencia a la forma en que se encuentran distribuidos los principales elementos que lo componen: plataformas, motores, ruedas. Existen distintas configuraciones, de las cuales tenemos: diferencial, triciclo, Ackerman, sincronizada,

omnidireccional, con múltiples grados de libertad y movimiento mediante orugas. **[10]**

3.4 CONFIGURACIÓN ACKERMAN

Es usado en la industria de automóviles ya que posee dos ruedas traseras de tracción y dos ruedas delanteras para la dirección. Esta configuración esta creada para evitar el derrape de las ruedas, haciendo que la rueda delantera interior posea un ángulo θi ligeramente mayor que el ángulo de la rueda exterior θo cuando el sistema se encuentra girando. Esto puede verse en el modelo cinemático mostrado en la figura 3.3.

Dónde:

L = separación lateral entre las ruedas.

D= separación longitudinal entre las ruedas.

ω= es la velocidad angular del robot.

v= velocidad lineal del robot

θ=ángulo de rotación del robot.

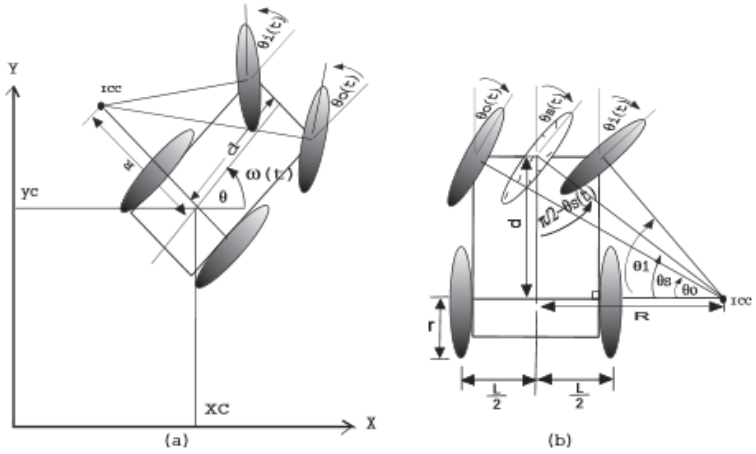


Figura 3.3 Modelo Cinemático Ackerman

La configuración Ackerman constituye un buen sistema de tracción, aplicado inclusive para terrenos inclinados.

3.5 ANÁLISIS MATEMÁTICO

El sistema Ackerman es una arquitectura utilizada actualmente en vehículos motorizados, su mecanismo de dirección une el volante con las ruedas delanteras. Con su ausencia resultaría muy difícil manejar un vehículo. A partir de la arquitectura Ackerman como se muestra en la Figura 3.4 se hace un análisis crítico que arroja una ecuación que relaciona el ángulo de giro y parámetros longitudinales físicos de la estructura.

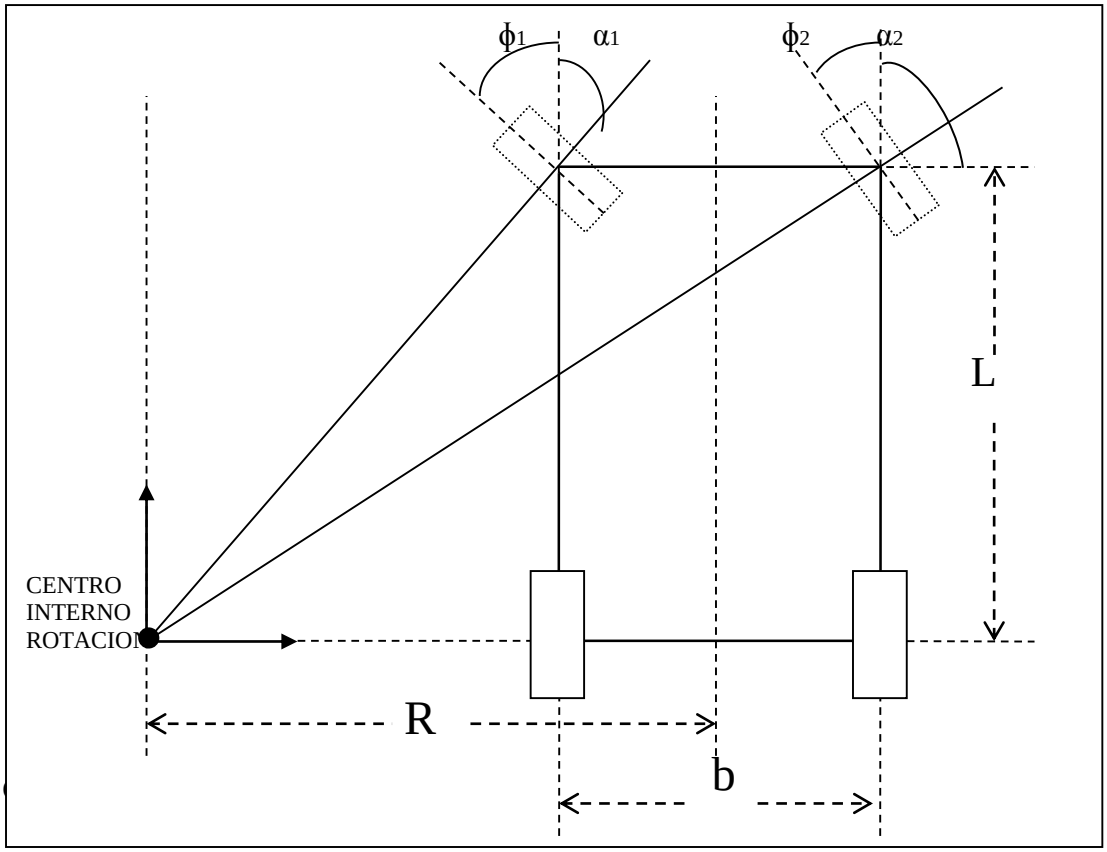


Figura 3.4 Arquitectura Ackerman

Parámetros:

L=Medida del largo del robot.

R=Medida del ancho del robot.

b=Radio de curvatura con respecto al eje de rotación.

ϕ1=Angulo de giro de rueda interna.

ϕ2= Angulo de giro de rueda externa.

Análisis:

α1 =90°-ϕ1 Ecuación (3.2)

$\mathrm{Tan}(\alpha 1) = \frac{R - \frac{b}{2}}{L}$ Ecuación (3.3)

$$\frac{sen(90°-\phi 1)}{cos(90°-\phi 1)} = \frac{R-\frac{b}{2}}{L}$$

$$\frac{cos(\phi 1)}{sen(\phi 1)} = \frac{R-\frac{b}{2}}{L}$$

$$ctg(\phi 1) = \frac{R-\frac{b}{2}}{L}$$

Haciendo una referencia obtendremos que:

$$ctg(\phi 1) = \frac{R-\frac{b}{2}}{L}$$

$$ctg(\phi 2) = \frac{R+\frac{b}{2}}{L}$$

$$ctg(\phi 2) - ctg(\phi 1) = \frac{b}{L}$$

CAPÍTULO 4

4. DISEÑO E IMPLEMENTACIÓN

En este capítulo se muestra el diseño de la minicomputadora y la electrónica de fuerza del robot móvil, además del acoplamiento de la tarjeta De0NANO con los periféricos externos como son: Módulo Bluetooth, Sensores Infrarrojos, Driver Puente H y motores DC.

4.1 DISEÑO EN SOFTWARE

Qsys (ver figura 4.1) es una herramienta utilizada para el diseño e integración de componentes utilizados para crear la minicomputadora. Se han agregado componentes: PWM, ADC y UART que ayudan en la interacción la tarjeta con dispositivos externos, por medio de una algoritmo creado en NIOS II.

Figura 4.1 Icono de Qsys

4.1.1 INCORPORACIÓN DE COMPONENTES EN QSYS

La interfaz de Qsys (ver figura 4.2) presenta una librería de componentes que son añadidos a la ventana de System Contents y estos componentes se configuran según los requerimientos del diseñador.

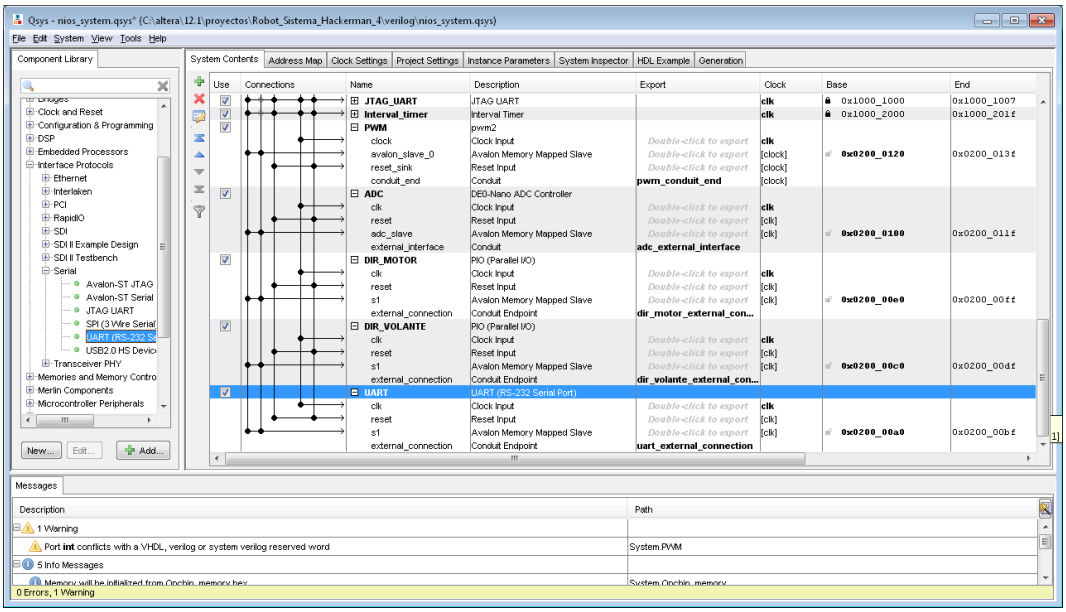


Figura 4.2 Ventana principal de Qsys del Quartus II de Altera

Se agrega un componente de PWM (ver figura 4.3), que permite generar señales de onda cuadrada para controlar la velocidad de los motores, por medio de la duración del ancho de pulso.

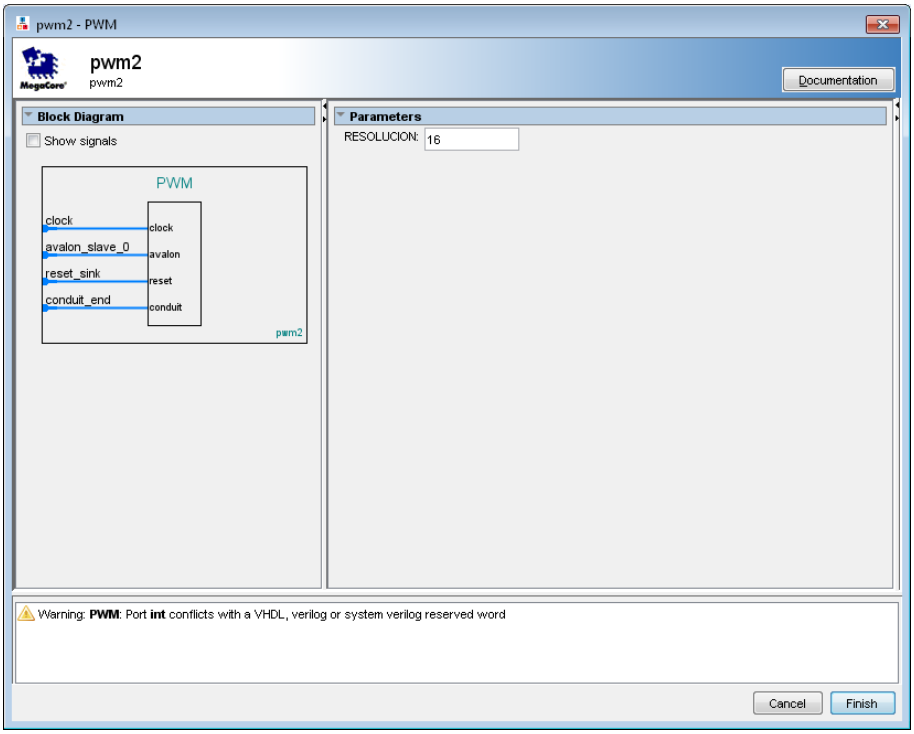


Figura 4.3 Configuración de PWM en Qsys

Para fijar el sentido y dirección de movimiento del robot se utilizó dos puntos paralelos configurados como salida de 2 bits como se muestra en la Figura 4.4.

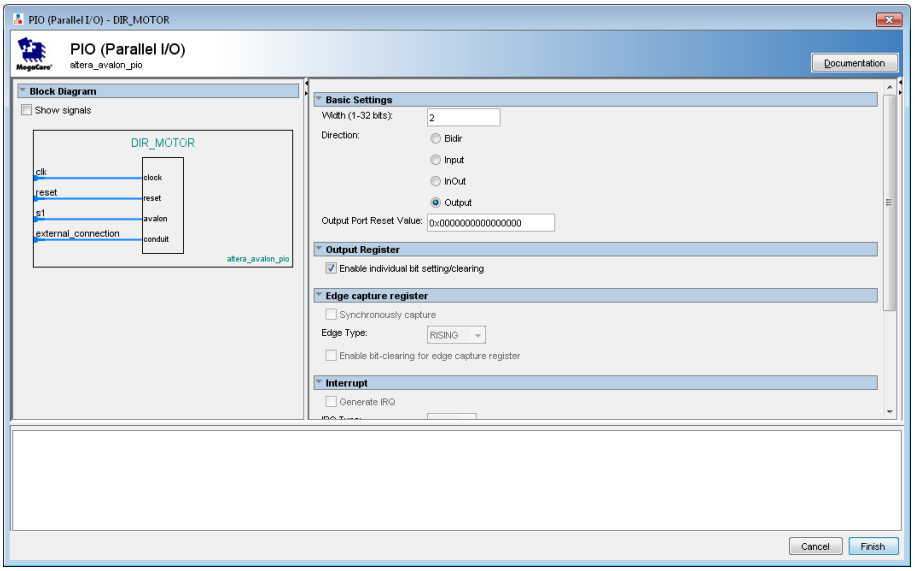


Figura 4.4 Configuración PIO para dirección de volante

Se añadió un componente ADC (ver figura 4.5), configurado con cuatro canales de lectura analógica asignado a cada sensor de distancia.

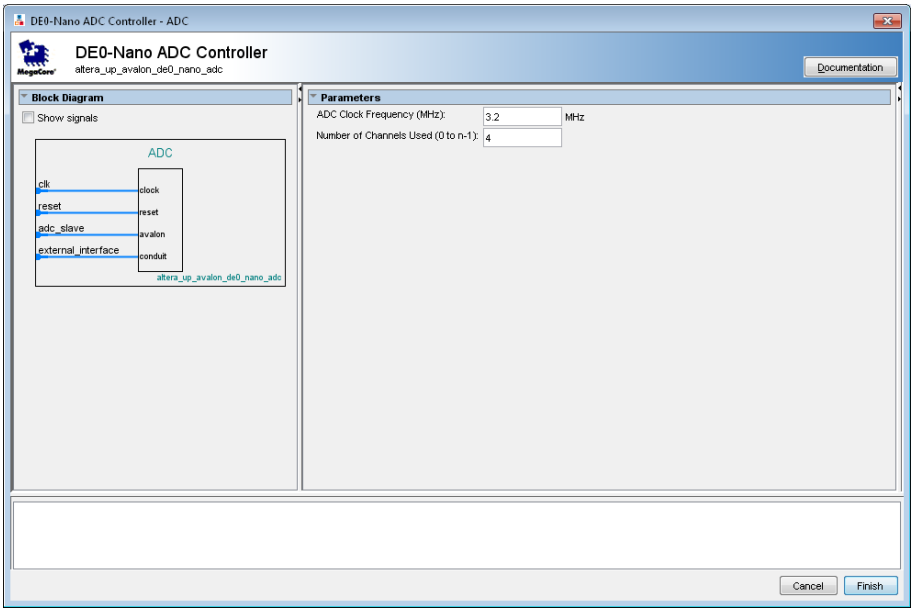


Figura 4.5 Configuración de ADC Controller

El enlace entre el dispositivo HC06 y la tarjeta de Altera se realiza a través de comunicación serial UART, por lo que se debe agregar un componente UART (RS-232 Serial Port) como se muestra en la figura 4.6.

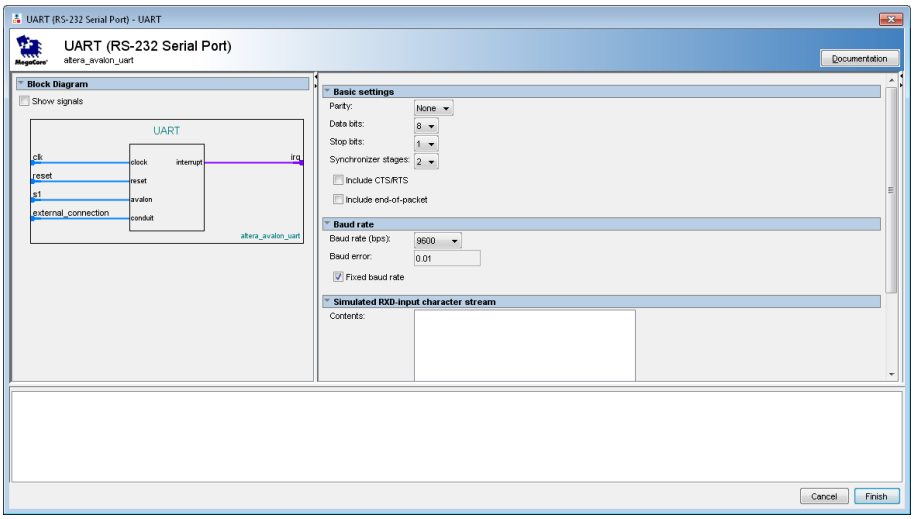


Figura 4.6 Configuración UART del Quartus II de Altera

En la Figura 4.7 se muestra todos los componentes utilizados en la máquina para el desarrollo del robot móvil.

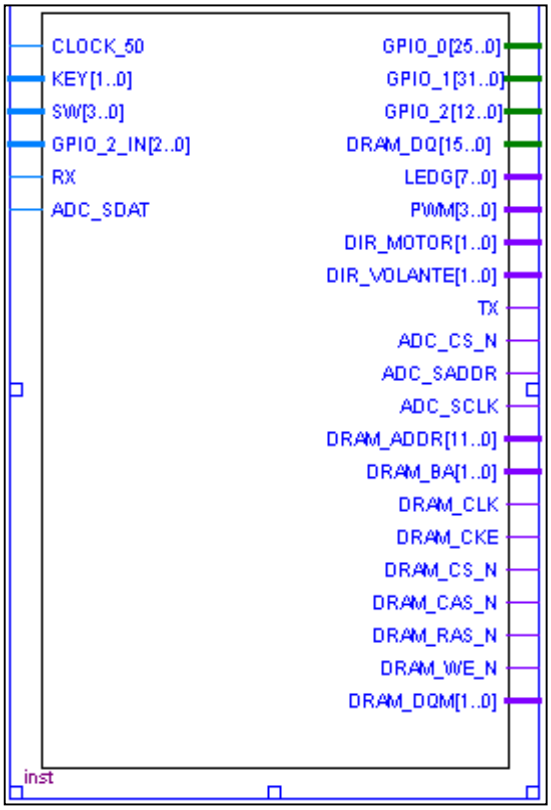


Figura 4.7 Diagrama Esquemático de la máquina

4.1.2 ENTORNO DE PROGRAMACIÓN

4.1.2.1 PROGRAMACIÓN EN NIOS II

MODO MANUAL

```
#include "altera_up_avalon_parallel_port.h"
#include "altera_up_avalon_de0_nano_adc.h"
#include "Sk_Pwms.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
```

```
#include <system.h>
void delay(int tiempo)
{
	int contador=0;
	while(contador!=tiempo)
	contador++;
}

void control(int * motor, int sentido,int vel_motor,int *
direccion,int giro,int vel_direccion)
{
	*(direccion)=giro;
	PWMS_SetDutyCycle(PWM_BASE,PWMS_CHANNEL_0,vel_direccion);
	*(motor)=sentido;
	PWMS_SetDutyCycle(PWM_BASE,PWMS_CHANNEL_1,vel_motor);
	//alt_up_parallel_port_write_data (led, 0);
}
int main (void)
{
	FILE* fp;
	fp = fopen ("/dev/UART","r+"); //Abre el archivo para leer y escribir
	alt_up_parallel_port_dev * led;
	alt_up_de0_nano_adc_dev * adc;
	volatile int * motor = (int *) DIR_MOTOR_BASE;
	volatile int * direccion = (int *) DIR_VOLANTE_BASE;
	unsigned int
data0,data1,data2,data3,velocidad,timon_izq,timon_der,velocidad_cont
rolada,dist_frente,dist_atras,dist_izq,dist_der,cobertura,vel_motor,
vel_direccion;
		unsigned int sentido,bandera,centinela;
		unsigned int
lectura_adc0_frente,lectura_adc1_atras,lectura_adc2_izquierda,lectur
a_adc3_derecha;
		char s1=0,s2=0,modo='1';
		int canal0,canal1,canal2,canal3,k;
		int derecha, izquierda, adelante, atras,parar;
```

```
data0  = 0;
data1  = 0;
data2  = 0;
data3  = 0;
canal0 = 0;
canal1 = 1;
canal2 = 2;
canal3 = 3;
dist_frente=0;
dist_atras=0;
dist_izq=0;
dist_der=0;
cobertura=50;
adelante=1;
atras=2;
izquierda=2;
derecha=1;
parar=0;
lectura_adc0_frente=0;
lectura_adc1_atras=0;
lectura_adc2_izquierda=0;
lectura_adc3_derecha=0;
//MODO MANUAL
velocidad=2300;
vel_motor=1700;
vel_direccion=2300;
//MODO INTELIGENTE
velocidad_controlada=1000;
timon_izq=1500;
timon_der=1500;
bandera=0;
k=0;
sentido=0;
centinela=0;
led =alt_up_parallel_port_open_dev ("/dev/Green_LEDs");
adc = alt_up_de0_nano_adc_open_dev ("/dev/ADC");
```

```
	alt_up_de0_nano_adc_update (adc);//probando
	PWMS_Init((void*)PWM_BASE,PWM_IRQ_INTERRUPT_CONTROLLER_I
	D,PWM_IRQ,10,2500);
	alt_up_parallel_port_write_data (led, 0xff)
	*(direccion)=izquierda;  //PARAR
	PWMS_SetDutyCycle(PWM_BASE,PWMS_CHANNEL_0,0);
	*(motor)=adelante;
	PWMS_SetDutyCycle(PWM_BASE,PWMS_CHANNEL_1,0);
	alt_up_parallel_port_write_data (led, 0);
	printf("HOLA");
	modo=1;
	alt_up_parallel_port_write_data (led, 255);
while(1)
{
	s1 = getc(fp);
	if (s1 =='1')                 //MODO SEMIAUTONOMO
	{
		modo='1';
		break;
	}

	if (s1 =='2')                 //IR A MODO MANUAL
	{
		modo='2';
		break;
	}
	if (s1 =='W')                 //ADELANTE
	{

control(motor,adelante,vel_motor,direccion,0,vel_direccion);
	alt_up_parallel_port_write_data (led, 1);
	}
	if (s1 =='S')                 //PARAR
	{
		alt_up_parallel_port_write_data (led, 2);
		control(motor,0,vel_motor,direccion,0,0);
```

```
}
if (s1 =='X')                //ATRAS
{
      control(motor,atras,vel_motor,direccion,0,0);
      alt_up_parallel_port_write_data (led, 4);
}
if (s1 =='Q')                //IZQUIERDA ADELANTE
{
      control(motor,adelante,vel_motor,direccion,izquier
      da,vel_direccion);
      alt_up_parallel_port_write_data (led, 8);
}

if (s1 =='E')                //DERECHA ADELANTE
{
control(motor,adelante,vel_motor,direccion,derecha,vel_d
ireccion);
      alt_up_parallel_port_write_data (led, 16);
}
if (s1 =='Z')                //IZQUIERDA ATRAS
{
control(motor,atras,vel_motor,direccion,izquierda,vel_di
reccion);
      alt_up_parallel_port_write_data (led, 32);
}

if (s1 =='C')           //DERECHA ATRAS
{
      control(motor,atras,vel_motor,direccion,derecha,vel_direccion);
      alt_up_parallel_port_write_data (led, 64);
}
if (s1 =='A')                //VOLANTE A LA IZQUIERDA
{
      control(motor,adelante,0,direccion,izquierda,vel_direccion);
      alt_up_parallel_port_write_data (led, 128);
}
```

```
            if (s1 =='D')                   //VOLANTE A LA DERERCHA
            {
                  control(motor,adelante,0,direccion,derecha,vel_direccion);
                  alt_up_parallel_port_write_data (led, 0);
            }
      }
return 0;
}
```

MODO SEMI-AUTONÓMO

```
#include "altera_up_avalon_parallel_port.h"
#include "altera_up_avalon_de0_nano_adc.h"
#include "Sk_Pwms.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <system.h>

void delay(int tiempo)
{
	int contador=0;
	while(contador!=tiempo)
		contador++;
}

void equilibrio(int * motor, int vel_motor,int * direccion,int
vel_direccion,int dist_frente,int adelante,int dist_der,int
dist_izq,int atras,int izquierda,int derecha,int adc,int canal0,int
data0, int canal2,int data2,int canal3, int data3)
{
	while(1)
	{
		// LECTURA ADC 0
	//alt_up_de0_nano_adc_update (adc);
	data0 = alt_up_de0_nano_adc_read (adc, canal0);
	dist_frente = (data0/16);

		if (dist_frente < 120)
		{
	control(motor,adelante,vel_motor,direccion,0,vel_direccion);

		}
		else
		{   //parar
```

```
control(motor,0,0,direccion,0,0);
            delay(200000);
// LECTURA ADC 3

//alt_up_de0_nano_adc_update (adc);
data3 = alt_up_de0_nano_adc_read (adc, canal3);
//dist_der = (data3/16)-100663296;
 dist_der = (data3-pow(2,29)-pow(2,30))/16;

if(dist_der < 120)
{
      while(dist_der <120 )
{
      control(motor,atras,vel_motor,direccion,izquierda,vel_direccion);
      // LECTURA ADC 3

data3 = alt_up_de0_nano_adc_read (adc, canal3);
//dist_der = (data3/16)-100663296;
dist_der = (data3-pow(2,29)-pow(2,30))/16;

}
//parar
control(motor,0,0,direccion,0,0);
delay(200000);
//derecha adelante
// LECTURA ADC 2

//alt_up_de0_nano_adc_update (adc);
data2 = alt_up_de0_nano_adc_read (adc, canal2);
//dist_izq = (data2/16)-67108864;
dist_izq = (data2-pow(2,30))/16;
while(dist_izq>120)
{
control(motor,adelante,vel_motor,direccion,derecha,vel_direccion);
```

```
      // LECTURA ADC 2
            //alt_up_de0_nano_adc_update (adc);
            data2 = alt_up_de0_nano_adc_read (adc, canal2);
            //dist_izq = (data2/16)-67108864;
            dist_izq = (data2-pow(2,30))/16;
                        }
                  }
            }
      }
}


void control(int * motor, int sentido,int vel_motor,int *
direccion,int giro,int vel_direccion)
{
      *(direccion)=giro;
      PWMS_SetDutyCycle(PWM_BASE,PWMS_CHANNEL_0,vel_direccion);
      *(motor)=sentido;
      PWMS_SetDutyCycle(PWM_BASE,PWMS_CHANNEL_1,vel_motor);
      //alt_up_parallel_port_write_data (led, 0);
}

int extraer_promedio(int canal,alt_up_de0_nano_adc_dev *adc)
{
int
inc=0,acum=0,prom=0,data0,data1,data2,data3,dist_frente,dist_atras,d
st_izq,dist_der;
if(canal==0)
{
   while(inc<19)
      {
            alt_up_de0_nano_adc_update (adc);
            data0 = alt_up_de0_nano_adc_read (adc,0);
            //dist_atras = (data1/16)-33554432;
            dist_frente = (data0/16);
            //printf("ADC1-ATRAS = %d \n",dist_atras);
```

```
        acum=acum+dist_frente;
        inc= inc+1;
        delay(5000);
            }
        prom=acum/20;
        inc=0;
        // LECTURA ADC 2
  }
  if(canal==1)
  {     while(inc<19)
           {
        alt_up_de0_nano_adc_update (adc);

        data1 = alt_up_de0_nano_adc_read (adc,1);
        //dist_atras = (data1/16)-33554432;
        dist_atras = (data1-pow(2,29))/16;
        //printf("ADC1-ATRAS = %d \n",dist_atras);
        acum=acum+dist_atras;
        inc= inc+1;
        delay(5000);
            }
            prom=acum/20;
  inc=0;
// LECTURA ADC 2
  }
  if(canal==2)
        {
              while(inc<19)
              {
              alt_up_de0_nano_adc_update (adc);
              data2 = alt_up_de0_nano_adc_read (adc,2);
              //dist_izq = (data2/16)-67108864;
              dist_izq = (data2-pow(2,30))/16;
              //printf("ADC2-IZQUIERDA = %d \n",dist_izq);
              acum=acum+dist_izq;
              inc= inc+1;
```

```
                    delay(5000);
                      }
                prom=acum/20;
                inc=0;
            }

// LECTURA ADC 3
if(canal==3)
      {           while(inc<19)
                  {
                        alt_up_de0_nano_adc_update (adc);
                        data3 = alt_up_de0_nano_adc_read (adc,3);
                        //dist_der = (data3/16)-100663296;
                        dist_der = (data3-pow(2,29)-pow(2,30))/16;
                        //printf("ADC3-DERECHA = %d \n",dist_der);
                        acum=acum+dist_der;
                        inc= inc+1;
                        delay(5000);
                        }
                              prom=acum/20;
                              inc=0;
            }
            return prom;
}

//---------------------Programa Principal-------------------------//
int main (void)
{
      FILE* fp;
      fp = fopen ("/dev/UART","r+"); //Abre el archivo para leer y escribir
      alt_up_parallel_port_dev * led;
      alt_up_de0_nano_adc_dev * adc;
      volatile int * motor = (int *) DIR_MOTOR_BASE;
      volatile int * direccion = (int *) DIR_VOLANTE_BASE;
```

```
unsigned int
data0,data1,data2,data3,velocidad,timon_izq,timon_der,velocidad_cont
rolada,dist_frente,dist_atras,dist_izq,dist_der,cobertura,vel_motor,
vel_direccion;
            unsigned int sentido,bandera,centinela;
            unsigned int
lectura_adc0_frente,lectura_adc1_atras,lectura_adc2_izquierda,lectur
a_adc3_derecha;
            char s1=0,s2=0,modo='1';
            int canal0,canal1,canal2,canal3,k;
            int derecha, izquierda, adelante, atras,parar;
            data0  = 0;
            data1  = 0;
            data2  = 0;
            data3  = 0;
            canal0 = 0;
            canal1 = 1;
            canal2 = 2;
            canal3 = 3;
            dist_frente=0;
            dist_atras=0;
            dist_izq=0;
            dist_der=0;
            cobertura=50;
            adelante=1;
            atras=2;
            izquierda=2;
            derecha=1;
            parar=0;
            lectura_adc0_frente=0;
            lectura_adc1_atras=0;
            lectura_adc2_izquierda=0;
            lectura_adc3_derecha=0;

//-----------------MODO MANUAL----------------------//
            velocidad=2300;
```

```
            vel_motor=1400;
            vel_direccion=1300;
//----------------- MODO INTELIGENTE -------------------//
            velocidad_controlada=1000;
            timon_izq=1500;
            timon_der=1500;
            bandera=0;
            k=0;
            sentido=0;
            centinela=0;
            //*(motor)=adelante;
            //*(direccion)=izquierda;

            led =alt_up_parallel_port_open_dev ("/dev/Green_LEDs");
            adc = alt_up_de0_nano_adc_open_dev ("/dev/ADC");
            alt_up_de0_nano_adc_update (adc);//probando

      PWMS_Init((void*)PWM_BASE,PWM_IRQ_INTERRUPT_CONTROLLER_ID,PWM_
IRQ,10,2500);

            alt_up_parallel_port_write_data (led, 0xff);

            *(direccion)=izquierda;  //PARAR
            PWMS_SetDutyCycle(PWM_BASE,PWMS_CHANNEL_0,0);
            *(motor)=adelante;
            PWMS_SetDutyCycle(PWM_BASE,PWMS_CHANNEL_1,0);
            alt_up_parallel_port_write_data (led, 0);
            printf("INICIANDO ADQUISICION");
            modo=1;
            alt_up_parallel_port_write_data (led, 0);//Apagar leds
      while(1)
      {
                  while(1)
                  {
```

```
      control(motor,adelante,vel_motor,direccion,derecha,vel_direcci
on);              }

      lectura_adc0_frente=extraer_promedio(canal0,adc);
      delay(10000);
      while(lectura_adc0_frente<120)//mientras no existe un obstaculo en
frente
      {
            lectura_adc3_derecha=extraer_promedio(canal3,adc);
            lectura_adc2_izquierda=extraer_promedio(canal2,adc);
            //CONDICIONES DE ROTACION

            if (lectura_adc3_derecha>80 && lectura_adc3_derecha<
            110)// contorno por derecha
            {
                  printf("GIRO DERECHA  \n\n");
                  if (lectura_adc3_derecha<120)// derecha
                        {
            control(motor,adelante,vel_motor,direccion,derecha,vel_direccion);
                        }
                  else
                        {
            if (lectura_adc3_derecha>125)// alejar de la derecha
      control(motor,adelante,vel_motor,direccion,izquierda,vel_direc
ion);
            else
      control(motor,adelante,vel_motor,direccion,0,0);//seguir recto
            }
      }
if (lectura_adc2_izquierda>80 && lectura_adc2_izquierda< 110)//
izquierda
      {
      printf("GIRO IZQUIERDA  \n\n");
      if (lectura_adc2_izquierda<120)//acercar a izquierda
            {
```

```
      control(motor,adelante,vel_motor,direccion,izquierda,vel_direc
cion);
            }
                  else
                  {
                  if (lectura_adc2_izquierda>125)// alejar de la
            izquierda
      control(motor,adelante,vel_motor,direccion,derecha,vel_direcci
on);
                  else
      control(motor,adelante,vel_motor,direccion,0,0);//seguir recto
                  }
            }

      lectura_adc0_frente=extraer_promedio(canal0,adc);
      printf("FRE = %d \n\n",lectura_adc0_frente);
      printf("IZQ = %d \n\n",lectura_adc2_izquierda);
      printf("DER = %d \n\n",lectura_adc3_derecha);
      }

      while(lectura_adc0_frente>=120)//IDENTIFICA HACIA A DONDE VA
      {
                        printf("Obstaculo  \n\n");

      lectura_adc0_frente=extraer_promedio(canal0,adc);
      control(motor,0,0,direccion,0,0);//parar
      delay(200000);

      lectura_adc0_frente=extraer_promedio(canal0,adc);
      if(lectura_adc0_frente<120)
      control(motor,adelante,vel_motor,direccion,0,0);
      else
      {
            lectura_adc3_derecha=extraer_promedio(canal3,adc);
```

```
      lectura_adc2_izquierda=extraer_promedio(canal2,adc);
      if(lectura_adc3_derecha<110)//chance por derecha
      {
            printf("RETRO_D  \n\n");
      control(motor,0,0,direccion,0,0);//parar
      delay(500000);printf("GIRO DERECHA  \n\n");
      delay(500000);
      control(motor,atras,vel_motor,direccion,0,0);//seguir recto
      delay(500000);
      control(motor,0,0,direccion,0,0);//parar
      delay(500000);
      delay(500000);
      control(motor,adelante,vel_motor,direccion,derecha,vel_direcci
on);
      delay(500000);
      }
      else
      {
            if(lectura_adc2_izquierda<110)//chance por derecha
            {
                  printf("RETRO_I  \n\n");
                  control(motor,0,0,direccion,0,0);//parar
                  delay(500000);
                  delay(500000);printf("GIRO IZQUIERDA  \n\n");
            control(motor,atras,vel_motor,direccion,0,0);//seguir
recto
                  delay(500000);
                  control(motor,0,0,direccion,0,0);//parar
                  delay(500000);
                  delay(500000);

control(motor,adelante,vel_motor,direccion,izquierda,vel_direccion);
                  delay(500000);
            }
            else
```

```
            {
                  control(motor,0,0,direccion,0,0);//parar

                  }
            }
      }

                        //centinela++;
       }
/*while(lectura_adc0_frente>=120)//hacia atras e inclinado hacia la
derecha
{
      lectura_adc0_frente=extraer_promedio(canal0,adc);
      lectura_adc3_derecha=extraer_promedio(canal3,adc);
      control(motor,atras,vel_motor,direccion,derecha,vel_direccion)
;//atras hasta obstáculo
      //centinela++;
}
/*while(centinela!=0)
{
      lectura_adc0_frente=extraer_promedio(canal0,adc);
      lectura_adc3_derecha=extraer_promedio(canal3,adc);
      control(motor,adelante,vel_motor,direccion,izquierda,vel_direc
cion);
            centinela--;
}
            centinela=0;
      }
return 0;
}
```

4.1.2.2 DIAGRAMA DE FLUJO

MODO MANUAL

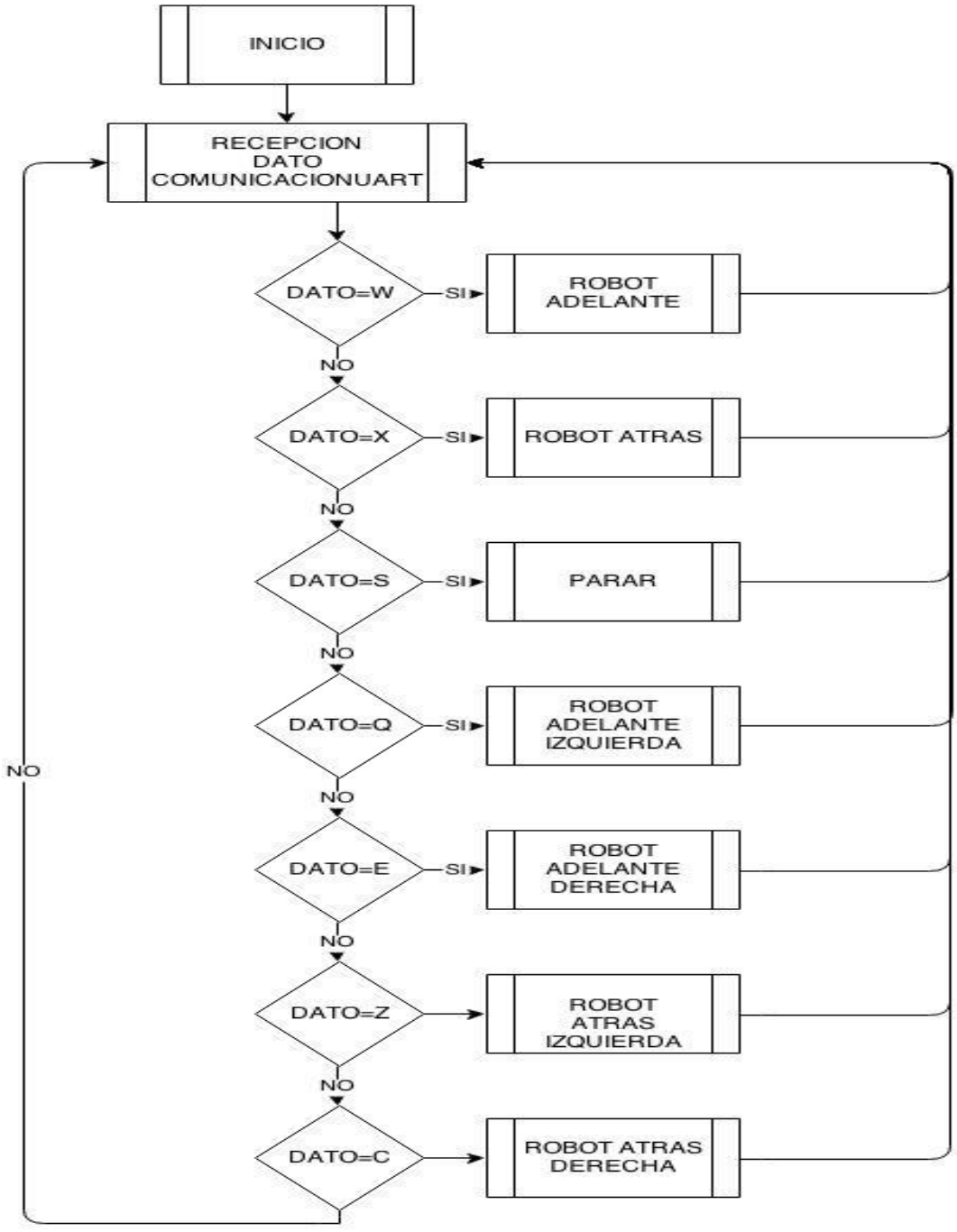


Figura 4.8 Diagrama ASM modo Manual

MODO SEMI-AUTONÓMO

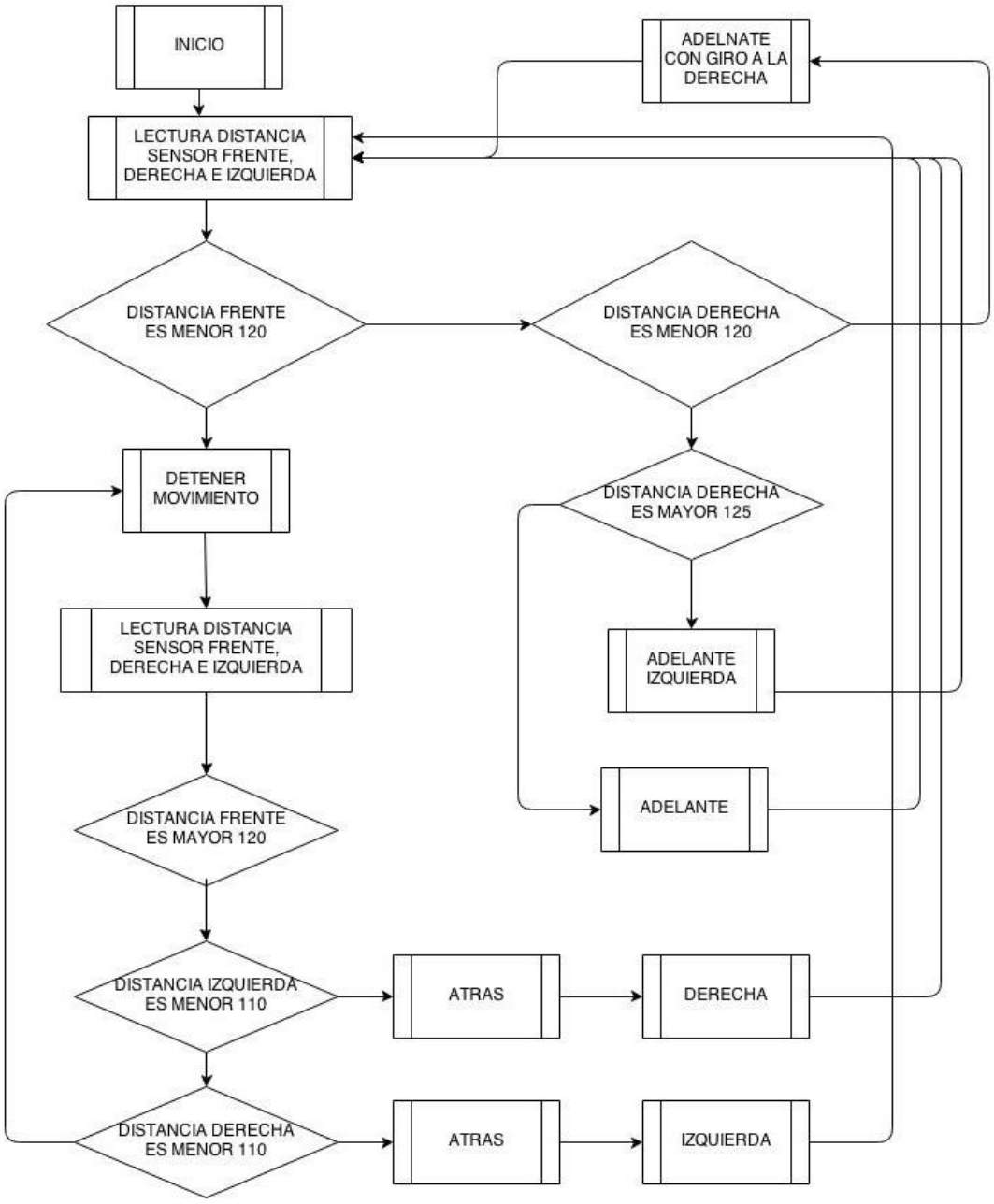


Figura 4.9 Diagrama ASM modo Semi - Autonomo

4.2 PROGRAMACIÓN EN APPINVENTOR

En la plataforma AppInventor se desarrolló la aplicación para controlar el robot. En la presentación inicial se define los bloques de algoritmo que inicializa la pantalla de móvil, el presionar el botón CONECTAR permite la comunicación directa vía Bluetooth como se muestra en la Figura 4.10.

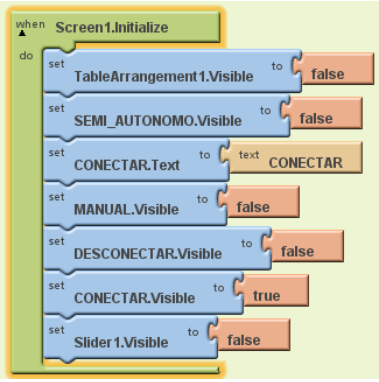


Figura 4.10 Bloques principales App Inventor

La Ver figura 4.11se muestra los bloques lógicos que permite al usuario visualizar la interfaz de control, previamente ya establecida el enlace entre el hardware y el dispositivo móvil.

Sino llegase a establecerse el enlace Bluetooth, la aplicación espera que se presione el botón CONECTAR.

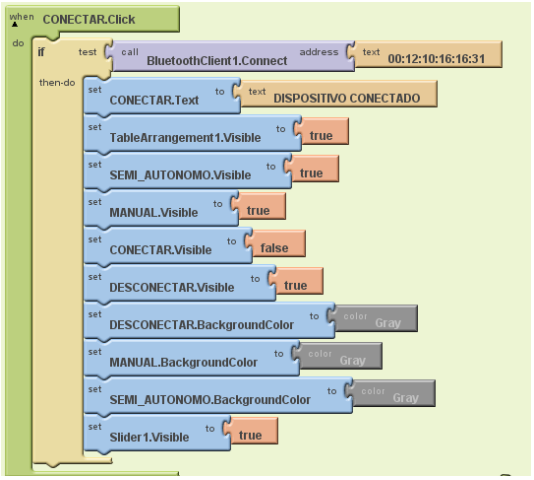


Figura 4.11 Desarrollo lógico de control App Inventor

Una vez establecido el enlace y la interfaz de control para el usuario. El canal permite el envío de datos como tramas de caracteres e interpretados de forma binaria (ver figura 4.12).

Figura 4.12 Bloque de control Bluetooth

4.2.1 SIMULADOR

Esta plataforma de desarrollo nos permite simular la presentación inicial como se muestra en la Figura 4.13 y las aplicaciones de control de robot móvil.

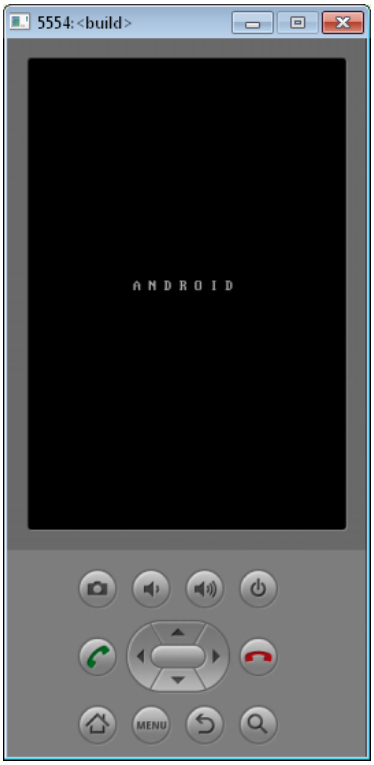


Figura 4.13 Simulador de Android

En la Figura 4.14 se muestra la simulación de la interfaz inicial.

Figura 4.14 Interfaz inicial de la aplicación

Como se apreciar (ver figura 4.15) las características del simulador es similar a la interfaz de los dispositivos móviles reales

Figura 4.15 Interfaz de control simulada.

Figura 4.16 Simulación Activa

4.3 IMPLEMENTACIÓN EN HARDWARE

Se presenta el prototipo electrónico diseñado que permite el control de forma remota del robot vía Bluetooth y la interconexión entre los dispositivos y la tarjeta De0NANO de Altera.

4.3.1 INTERFAZ ANDROID Y TARJETA DE0 NANO

Dispositivo Bluetooth que utiliza comunicación serial UART y actúa como esclavo, recibe los datos enviados desde el Smartphone.

DE0 NANO		BLUETOOTH	
RX	PIN_J14	DTX	PIN 2
TX	PIN_13	DRX	PIN 3

Tabla 4.1 Interconexión DE0 Nano y Modulo Bluetooth

Utilizamos el protocolo comunicación serial como medio de enlace vía Bluetooth para enviar la trama de datos. Para control del robot vía Bluetooth se usa el protocolo UART, debido a que el dispositivo móvil utiliza este medio para transmitir los datos

4.3.2 INTERFAZ SENSORES Y TARJETA DE0 NANO

Se utilizó sensores de distancia para detectar muros u obstáculos que impidan el paso del vehículo, estos sensores poseen un transductor, que envían un voltaje proporcional a la distancia medida. Y estos son de acuerdo a los datos obtenidos el robot toma decisiones para evitar colisiones.

DE0 NANO	SENSOR GP2D12		ESTADO
Analog_In0	Sensor frente	Pin_VO	Implementado
Analog_In1	Sensor Derecha	Pin_VO	Implementado
Analog_In2	Sensor Izquierda	Pin_VO	Implementado

Tabla 4.2 Interconexión DE0 Nano y sensores de distancia

4.3.3 INTERFAZ PUENTE H Y TARJETA DE0 NANO

Es un dispositivo que hará posible que este robot se mueva de un sector a otro, para esto se utilizó el integrado LM298 para proveer fuerza a los motores a su desplazamiento.

Cuando se habla de control de velocidad de motores queremos hacer notar que el PWM siempre estará vinculado a este sistema, debido a que se puede controlar la velocidad del móvil desde 0% hasta un 100%

DE0 NANO	PUENTE H
PIN_D11	DIR_VOLANTE_1
PIN_A12	DIR_VOLANTE _0
PIN_B12	DIR_MOTOR_1
PIN_D12	DIR_MOTOR_0
PIN_E11	PWM_0
PIN_E10	PWM_1

Tabla 4.3 Interconexión DE0 Nano y Puente H.

CAPÍTULO 5

5 PRUEBAS Y RESULTADOS

5.1 ANÁLISIS DE VELOCIDADES CON RESPECTO AL PWM.

En la Tabla 5.1 se representa la velocidad del móvil con respecto a su velocidad máxima. La velocidad máxima depende del suministro de corriente de las baterías y un PWM al 100%.

Los resultados muestran el aumento de velocidad con respecto al incremento de PWM.

La masa total del robot fue de 1.2Kg y un peso de 11.76 N.

PWM (%)	VELOCIDAD (m/s)	DISTANCIA (m)	TIEMPO (seg)
100	0,74	2	2,67
90	0,68	2	2,91
80	0,62	2	3,22
70	0,54	2	3,67
60	0.48	2	4,1
50	0,39	2	5,02
40	0,37	2	5,03

Tabla 5.1 Velocidad del móvil

5.2 DATOS EXPERIMENTALES DE LOS SENSORES

A continuación se obtuvieron los datos que representan las medidas de distancia dadas por el sensor GP2D12 y ubicadas en la Tabla 5.3. Se comparan sus respectivos valores experimentales con respecto a los valores teóricos dados por el fabricante.

DISTANCIA (cm)	VALOR TEORICO VOLTAJE [V]	VALOR EXPERIMENTAL VOLTAJE [V]	%ERROR
30	0,91	0,90	0,5
25	1,10	1,15	2,2
20	1,27	1,40	4,8
15	1,60	1,70	3,0
10	2,09	2,45	7,9

Tabla 5.2 Medidas obtenidas sensor GP2D12

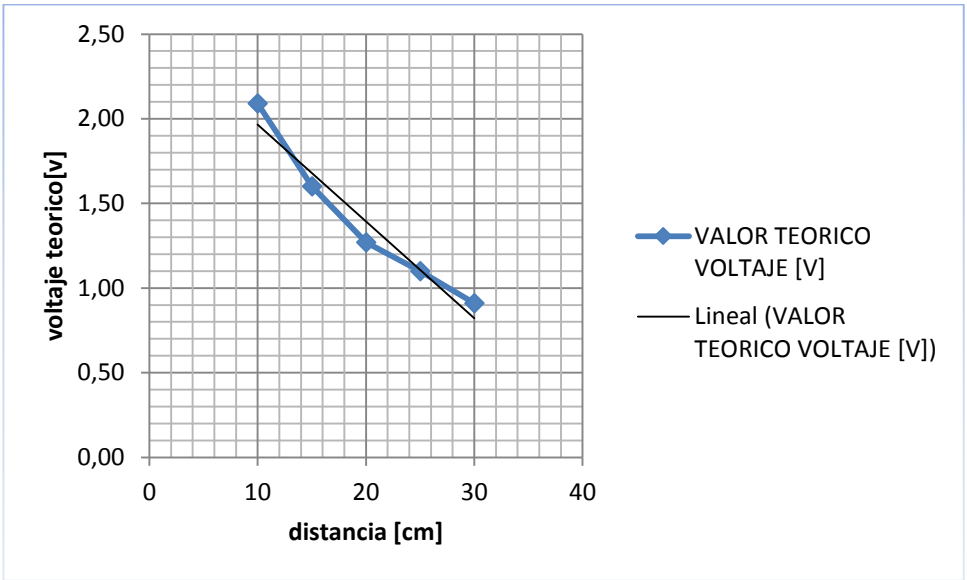


Figura 5.1 Relación Voltaje teórico Vs distancia

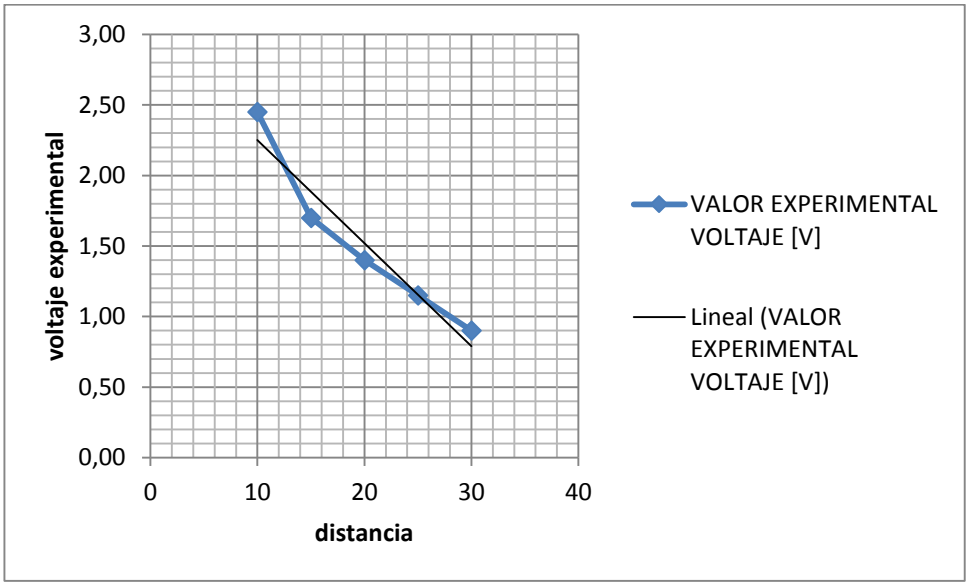


Figura 5.2 Relación Voltaje experimental Vs distancia

5.3 ANÁLISIS DEL MECANISMO DE GIRO SISTEMA ACKERMAN.

PWM [%]	Diámetro(m)	Angulo de giro máximo	Tiempo de recorrido (s)
100	2,25	20°	6,50
80	2,26	20°	7,20
60	2,37	20°	7,80
50	2,43	20°	7,83

Tabla 5.3 Datos experimentales de giro del robot

5.3.1 CÁLCULO MATEMÁTICO.

$$ctg(\phi 1) = \frac{R - \frac{b}{2}}{L}$$

L=Medida del largo del robot.

R=Medida del ancho del robot.

b=Radio de curvatura con respecto al eje de rotación.

ϕ1=Angulo de giro de rueda interna.

Datos obtenidos del prototipo:

b = 20 [cm]

L = 37 [cm]

$\phi 1 = 20°$

Desarrollo:

$$ctg(\phi 1) = \frac{R - \frac{b}{2}}{L}$$

$$R = ctg(\phi 1)\text{xL} + \frac{\text{b}}{2}$$

$$R = ctg(20)\text{x}(37) + \frac{20}{2}$$

Radio de giro aproximado = 111,665cm

Diámetro total aproximado = 223,33cm

	Valor teorico [cm]	Valor Experimental[cm]	Error [%]
Diametro	225,45	223.33	0,47

Tabla 5.4 Comparación entre diámetros.

5.4 REPRESENTACION DE DATOS EN ENLACE DE TELECONTROL

Los datos enviados para el telecontrol del robot con Sistema Ackerman fueron escogidos por motivo de posibles mejoras a futuro y ser controlado por medio de un computador con Bluetooth, los caracteres se ordenan para una cómoda manipulación de envío de comandos desde el teclado.

Estos caracteres son recibidos por el módulo Bluetooth y son interpretados por el robot.

Dirección	Caracter de control enviado.	Dato de Control formato Hexadecimal	Dato de control formato Binario
Adelante	W	0X57	01010111
Atrás	X	0X58	01011000
Giro de llantas a la derecha	D	0X44	01000100
Giro de llantas a la izquierda	A	0X41	01000001
Adelante e izquierda	Q	0X51	01010001
Adelante y derecha	E	0X45	01000101
Atrás e izquierda	Z	0X5A	01011010
Atrás y derecha	C	0X43	01000011
Incremento de cobertura sensor	4	0X34	00110100
Decremento de cobertura sensor	3	0x33	00110011

Tabla 5.5 Datos de control de dirección del robot.

Baterias BL5C	4	20
Modulo Bluetooth	1	40
Carcasa Robótica	1	70
Tablet Android	1	250
Sensor distancia	3	75
motores Dc	2	20
Total		630

Tabla 5.6 Lista de precios de componentes del proyecto

5.5 CONSUMO DE CORRIENTE TOTAL

DISPOSITIVO	CANTIDAD	CONSUMO MÍNIMO [mA]	CONSUMO MÁXIMO [mA]
DE0 NANO	1	10	20
L298N	1	12	36
SENSOR	3	90	120
BLUETOOTH	1	8	40
TOTAL		120	216

Tabla 5.7 Consumo de corriente de dispositivos.

CORRIENTE [mA]	TIEMPO [segundos]
80	25
77	30
75	60
74	73
72	112

Tabla 5.8 Consumo de corriente de baterías motor trasero

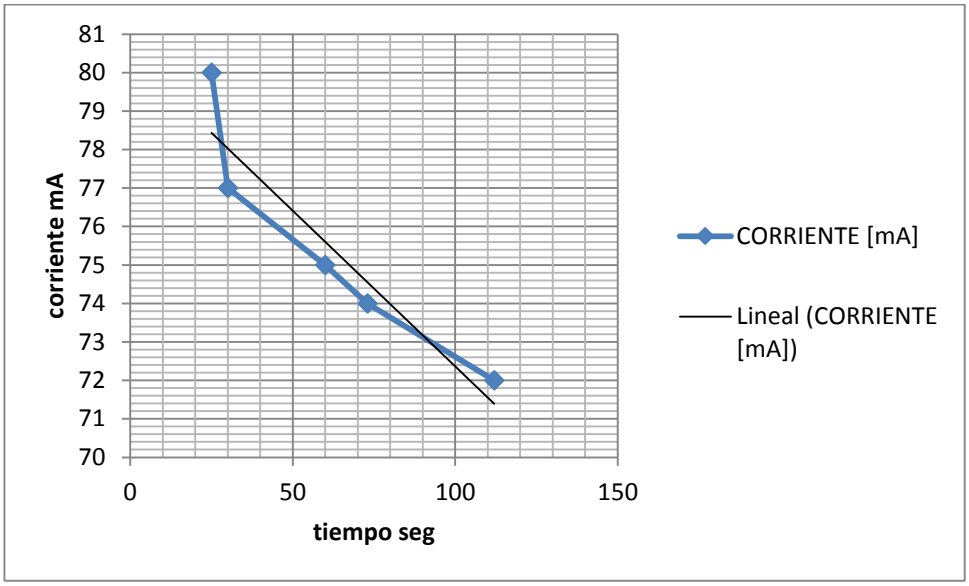


Figura 5.3 Tiempo vs Corriente (a)

CORRIENTE [mA]	TIEMPO [minutos]
90	2,14
89	2,24
88	2,30
87	3,09
85	3,20
84	3,40
83	3,55

Tabla 5.9 Consumo de corriente de baterías motor trasero y delantero

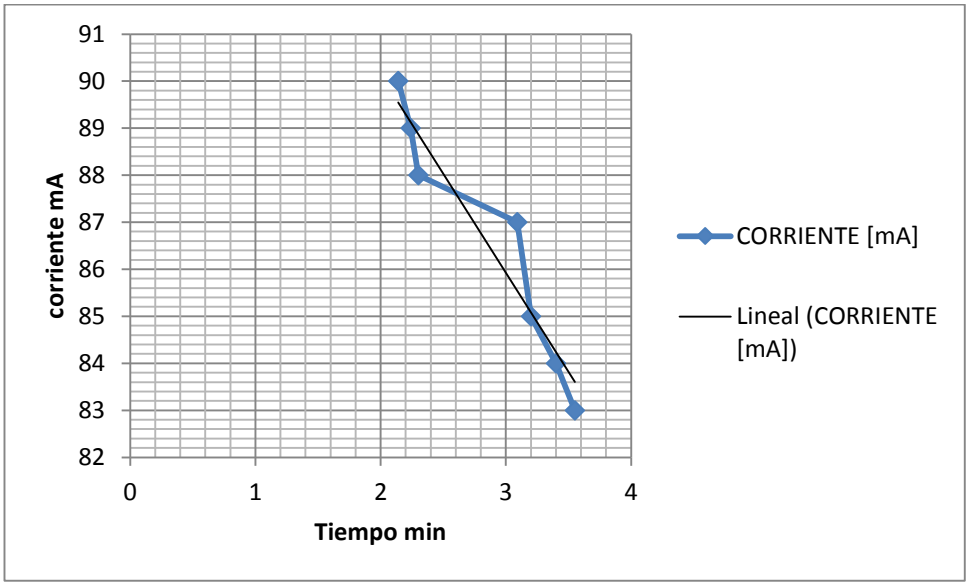


Figura 5.4 Tiempo vs Corriente (b)

CONCLUSIONES

1. La minicomputadora fue consolidada mediante la interconexión de hardware y periféricos externos, por medio de los interfaces agregadas en Qsys.

2. El consumo de corriente de los dispositivos afectan directamente a la funcionalidad de los sensores de distancia infrarrojos.

3. AppInventor disminuye el tiempo de diseño de aplicaciones Android y la curva de aprendizaje en programación de dispositivos móviles.
4. El enlace inalámbrico entre el dispositivo Android y el robot puede ser interrumpido por bajo voltaje en las baterías.

RECOMENDACIONES

1. Al momento de la adquisición de datos, es recomendable tomar un conjunto de muestras y promediarla, para ubicar el valor real aproximado, esto disminuirá los errores de ejecución.

2. Uno de los aspectos importantes a denotar es la funcionalidad de los sensores, la adquisición de datos depende del suministro adecuado de corriente de la fuente, ya que la corriente máxima requerida por cada sensor es de 33mA, con voltajes entre (4.5 – 5 voltios).

Por lo que se recomienda realizar análisis previo de eficiencia de trabajo ya que la calidad de alimentación que se suministre está directamente relacionada con la estabilidad del sistema.

3. Para realizar la lectura de datos que relacionen conversión analógica-digital, se recomienda analizar el patrón numérico existente para cada canal, y recalcular dichos valores, en rangos normalizados entre 0 - 255.

4. Los sensores infrarrojos utilizados, poseen un rango de distancia de máximo 80 cm, se recomienda utilizar un sensor ultrasónico que pueden detectar objetos a una distancia máxima de 4 metros y así el robot tenga distancia apropiada para evadir obstáculos.

5. Se recomienda utilizar un módulo Bluetooth de clase 1, para obtener mayor cobertura de alcance.

6. Es recomendable dividir las fuentes de alimentación mediante etapas, en nuestro caso control y fuerza; esta última es la encargada del manejo de motores. La importancia de tratar por separado estos bloques se debe a que el consumo de corriente en la sección de fuerza es mucho mayor, lo cual ocasionaría problemas al momento

de adquirir datos, ya que las fluctuaciones de corrientes podrían ocasionar cálculos erróneos por parte de la etapa de control.

BIBLIOGRAFÍA

- [1] Altera Corporation-DE0-Nano http://www.altera.com/education/univ/materials/boards/de0- Nano/unv de0-nano-board.html. (consultado el 10 de junio 2013).

- [2] Altera Corporation - University Program “Quartus_II_Simulation QSIM” (consultado el 10 de junio 2013).

- [3] Altera Corporation – “Embedded Design” , January 2011 http://www.altera.com/literature/hb/nios2/edh_ed_handbook.pdf

- [4] Vahid, F. & Givargis, T. “Embedded System Design”. John Wiley & Sons, Inc. 2002.
  Disponible en: http://dsp-book.narod.ru/ESDUA.pdf

- [5] Wang, C. “A Survey of Embedded Operating System”. 2005. Disponible:http://cseweb.ucsd.edu/classes/fa01/cse221/projects/group 2.pdf (consultado el 2 de Julio 2013).

- [6] David A. Perez A. - Centro de Investigación en Comunicación y Redes (CICORE), Caracas, Octubre 2009, “Sistemas Embebidos y Sistemas Operativos Embebidos” ND 2009-03

- [7] Rodríguez Araujo Jorge - “Estudio del Microprocesador Nios II”. Disponible en: http://es.scribd.com/doc/28358833/Estudio-del-microprocesador-Nios-II. Publicado Marzo 2010

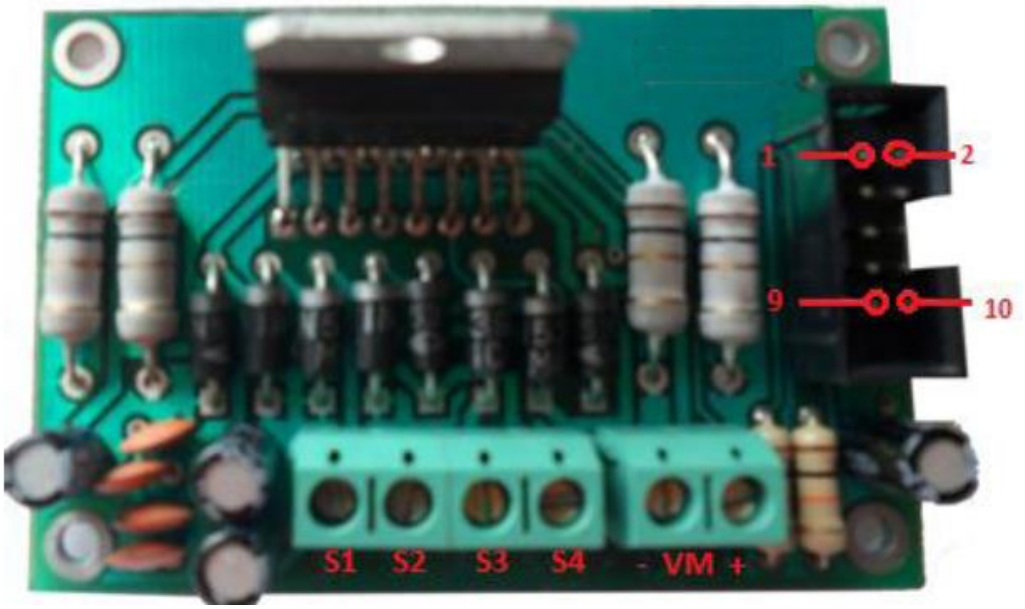
- [8] Altera Corporation, “Nios II Software” Disponible en: http://www.altera.com/literature/hb/nios2/n2sw_nii5v2.pdf (Consulta 6 de agosto del 2013)

- [9] Muller , Nathan J, “Tecnología Bluetooth ”, Madrid: McGraw-Hill,2002.

- [10] BOLTON. W. ,“Mecatrónica sistemas de control electrónico en ingeniería mecánica y eléctrica”, Editorial Alfaomega. Tercera edición. México, Febrero 2006

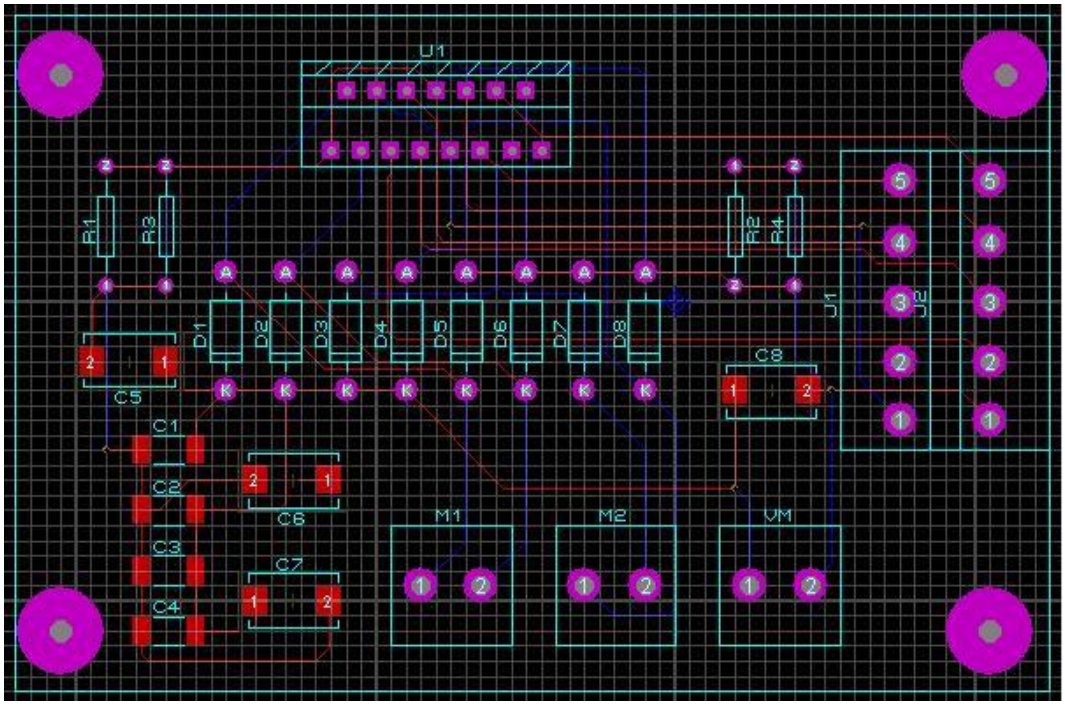
ANEXOS

Especificaciones Puente H.
S1-S2 : Motor de atrás.
S3-S4: Motor de adelante.
VM: Voltaje de alimentación de motores.



Pin_01: Entrada Dir_Motor_0.
Pin_02: Entrada PWM_0
Pin_03: Entrada Dir_Motor_1.
Pin_04: Entrada PWM_1
Pin_05: Entrada Dir_Volante_0.
Pin_06: NC.
Pin_07: Entrada Dir_Volante_1.
Pin_08: NC.
Pin_09: Vcc= 5 voltios.
Pin_10: GND.

PCB Puente H



Vista Puente H 3D