ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Wii Remote with LabVIEW"

Tesina de Seminario

Previa a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:
María Fernanda Ayerve Bermúdez
Evelyn del Rocío Rodríguez Alarcón

GUAYAQUIL – ECUADOR
AÑO
2011

A G R A D E C I M I E N T O

Quiero expresar mi gratitud a Dios, a mí madre y a mis tíos: Cecilia y Daniel por los valores que me supieron inculcar y por su ejemplo de perseverancia.

A mis hermanas, familiares y amigos, porque sé, que siempre puedo contar con ellos.

A Jair, mi novio y compañero del camino, por su apoyo incondicional.

Al Ing. Luis Vásquez por brindarme sus consejos, experiencia, por su paciencia y la dedicación que me ofreció en la realización de la tesis.

María Fernanda Ayerve Bermúdez

Quiero dar gracias a Dios, por estar conmigo en cada paso que doy, por fortalecer mi corazón e iluminar mi mente y por haber puesto en mi camino a aquellas personas que han sido mi soporte y compañía durante todo el período de estudio.

Agradecer hoy y siempre a mi familia en especial a mis padres por haberme enseñado el valor de los estudios y darme el apoyo suficiente en los momentos difíciles de mi carrera.

Al Ing. Luis Vásquez por la oportunidad brindada para la realización de la tesis, agradecerle por su paciencia, consejos experiencia y sobre todo su amistad.

Evelyn del Roció Rodríguez Alarcón

D E D I C A T O R I A

A Dios,

A mi madre,

A mis tíos,

A mis hermanas,

A mi novio.

María Fernanda Ayerve Bermúdez

A Dios,

A mis Padres,

A mi abuelita,

A mi hermana,

Evelyn del Roció Rodríguez Alarcón

TRIBUNAL DE SUSTENTACIÓN

Ing. Washington Medina
DELEGADO DECANO FIEC

Ing. Luis Fernando Vásquez Vera
PROFESOR SEMINARIO
DE GRADUACION

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Proyecto de Grado, nos corresponde exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

(Reglamento 4256 TITULO IV Capítulo II Art. 18 literal c)

_____

María Fernanda Ayerve Bermúdez

_____

Evelyn del Rocío Rodríguez Alarcón

RESUMEN

El objetivo de este proyecto es explorar las posibilidades del Wii Remote como interfaz de control, considerando también su uso como herramienta docente.

El proyecto está estructurado en 4 capítulos. En el primero de ellos se presentan las características, componentes electrónicos y especificaciones técnicas del mando Wii Remote, la tecnología Bluetooth y el software LabVIEW.

A continuación, en el segundo capítulo, se muestra un esquema general del proyecto mediante un diagrama de bloques detallando de manera breve la función del mismo.

En el tercer capítulo, se muestra el entorno de programación LabVIEW, la explicación detallada de cada uno de los SubVI's usados y creados.

Por último, en el cuarto capítulo, se da una visión del proceso de modificación del carro de juguete que de aquí en adelante nos referiremos a él como "mecanismo teleoperado", la interacción de éste con LabVIEW y el mando Wii Remote.

Finalmente se incluye un conjunto de referencias seleccionadas, principalmente páginas Web, que contienen información relevante para el desarrollo de este proyecto y de otros trabajos que puedan abordarse en el futuro.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En este proyecto titulado “Wii Remote with LabVIEW”, se propone el uso del mando inalámbrico Wii Remote, de la consola Wii de Nintendo, que junto con el software LabVIEW nos ayudan en la transmisión y recepción de los datos hacia un mecanismo teleoperado determinado.

Aquí, utilizaremos las capacidades con las que, los diseñadores de Nintendo han dotado al Wii Remote, para después proponer una herramienta de software que permita la interacción entre este y un PC.

Además, dado su carácter autónomo, no se requiere la adquisición de una consola Wii para su utilización conjunta con aplicaciones de PC, por lo que primero debemos conocer a detalle el funcionamiento de cada periférico del mando para proceder a aplicarlo en este proyecto.

A su vez, este proyecto hace uso de la tecnología Bluetooth y HID (Human Interface Device); por lo que debemos conocer sus funciones para así realizar las conexiones respectivas, pudiendo de esta manera transmitir voz, datos e imágenes, si así se lo desea.

CAPITULO 1

FUNDAMENTACIÓN TEÓRICA

1.1 Control de mando Wii Remote

El mando Wii Remote es aquel que tiene la característica de ser un dispositivo inalámbrico y de interfaz humana (HID); además su diseño simétrico le permite ser utilizado tanto con la mano izquierda como con la mano derecha por cualquier individuo con total libertad.

Figura 1.1 Mando Wii Remote

1.2 Partes que conforman el mando Wii Remote

El control Wii Remote está físicamente conformado por:

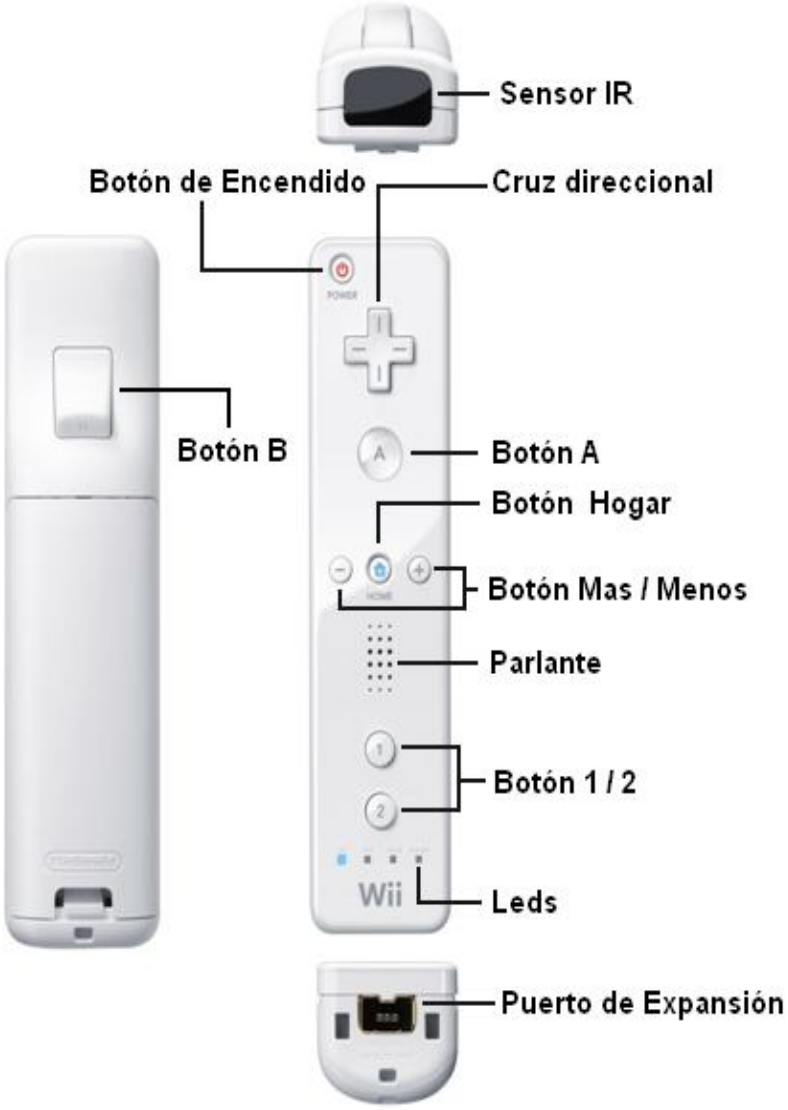


Figura 1.2 Partes que componen al mando Wii Remote

- Cámara infrarroja: para detección de movimientos del usuario.
- Botones (Encendido, A, - , Hogar, +, 1, 2 y B como gatillo en la parte inferior, botón de sincronización).
- Cruz direccional.
- Altavoz o parlante.
- 2 pilas AA: con una duración de entre 30 a 60 horas, según el uso o no de la función de puntero.
- 4 Leds: Indican el número de jugador o el nivel de batería.
- Puerto de expansión: para la conexión de otros elementos como el Nunchuck.

El control de Wii **internamente** tiene lo siguiente:

- Memoria EEPROM.
- Acelerómetro de 3 ejes.
- Módulo Bluetooth.
- Vibrador.

1.3 Descripción breve de los componentes del Wii Remote

1.3.1 Cámara infrarroja

Ubicada en la parte superior del Wii Remote, se encuentra una cámara monocroma (esta cámara solo puede detectar luz infrarroja o la ausencia de la misma) de 128x96 píxeles, similar a una webcam, pero sensible a luz infrarroja. Esta tiene un procesador incorporado, capaz de seguir hasta 4 objetos móviles.

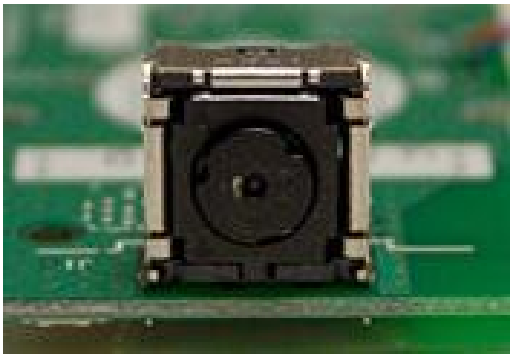


Figura 1.3 Wii Remote Cámara Infrarroja (IR)

La cámara puede trabajar en tres estados diferentes:

- Cámara en estado de no recogida de datos.
- Cámara recogiendo datos con una sensibilidad media.
- Cámara recogiendo datos con sensibilidad alta.

Cuando esta sensibilidad se reduce, también se reduce a su vez, la resolución del subpíxel, acercándose a la verdadera resolución del sensor (128x96 píxeles).

1.3.2 Botones

El “Wii Remote” está conformado por 10 botones distribuidos por toda su superficie, además del botón B, situado en la parte posterior del mando a modo de gatillo.

Al retirar la tapa de las baterías del Wii Remote situado en un lateral de las pilas encontraremos un botón de sincronización como se observa en la figura 1.4. Este botón es el que nos va a ayudar a sincronizar el Wii Remote con la PC, si surgiera algún problema.

Figura 1.4 Botón de Sincronización

Existen diferencias físicas en cuanto al tipo de botones, algunos basan su funcionamiento en un interruptor de tipo membrana (Cruz direccional, A, B, 1 y 2), mientras que otros se basan en pulsadores (POWER, -, HOME, +), (ver figura 1.11).

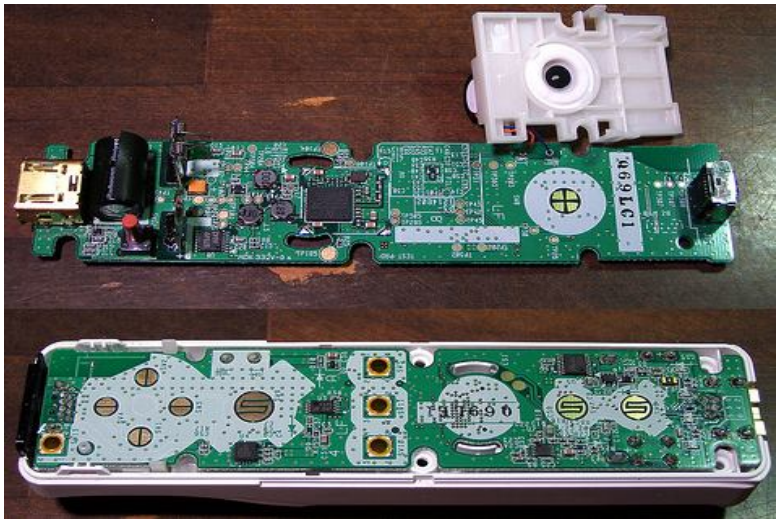


Figura 1.5 Circuito Impreso

1.3.3 Memoria EEPROM

La memoria se encuentra dividida en diferentes partes: memoria accesible para el usuario, registros de control y los registros de entrada.

El chip de 16KB EEPROM dispone de una sección de 6KB, en la cual el usuario puede leerla y escribirla de forma libre. En esta parte de la memoria es posible registrar diferentes perfiles de usuario.

Los registros de control hacen referencia a los diferentes periféricos incluidos en el mando, tales como:

- El altavoz.
- Los llamados controles de extensión, como por ejemplo el Nunchuck.
- La cámara de infrarrojos.

1.3.4 Acelerómetro de 3 ejes

Un acelerómetro es un dispositivo electrónico que permite medir vibraciones, mediante la medida de la aceleración de la gravedad.

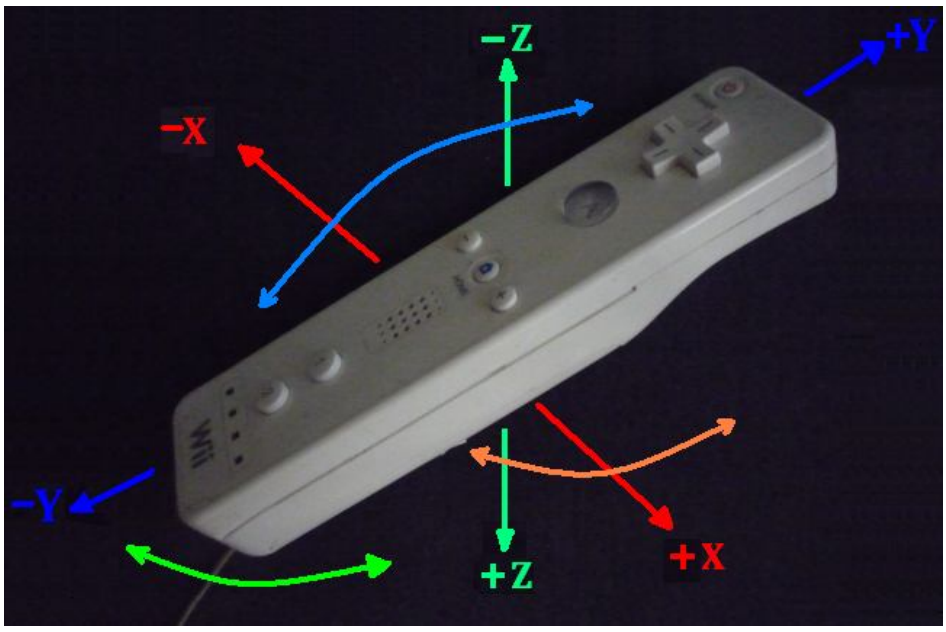


Figura 1.6 Ejes coordenados del mando

Este dispositivo se conoce como MEMS por sus siglas en inglés (Sistema Micro Electro-Mecánico), ya que convierte un movimiento mecánico en un valor eléctrico; es decir, la inclinación a un movimiento del mando en cualquiera de los tres ejes, es convertido en una diferencia de potencial.

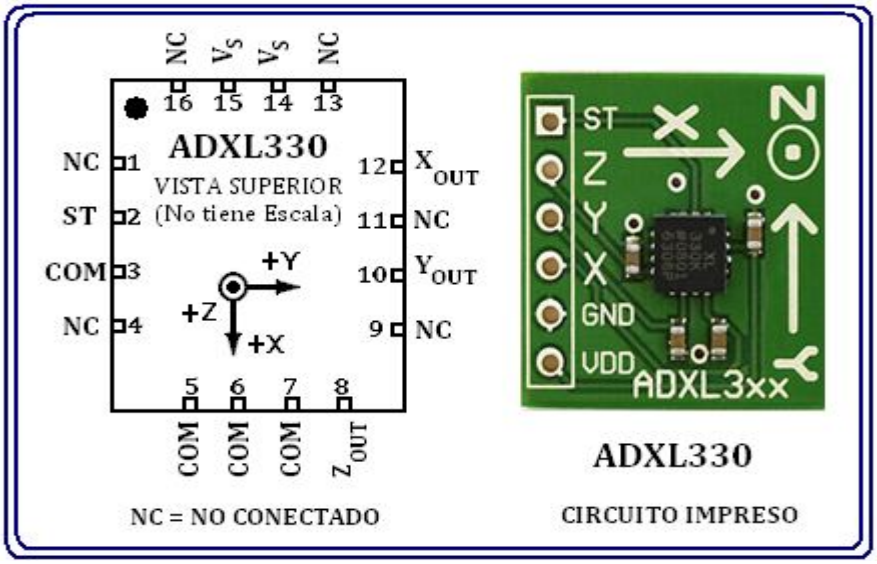
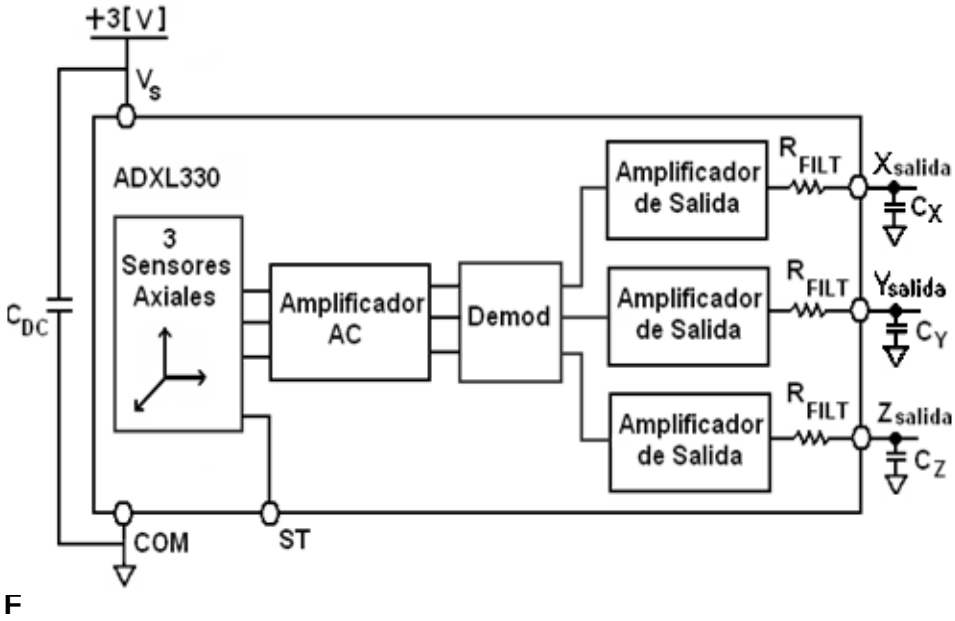


Figura 1.7 Chip MEMS ADXL330

El chip MEMS utilizado en el diseño electrónico del Wii Remote es el ADXL330, desarrollado por la multinacional Analog Devices. Las dimensiones del dispositivo electrónico son pequeñas (4 x 4 x 1,5 [mm]).

El rango de detección del acelerómetro abarca +/- 3 [g]. Además dispone de una sensibilidad típica de 300 [mV / g], siendo g = 9.81 [m/s^2], donde g es la aceleración de la gravedad.

En el esquema de la Figura 1.8, se aprecia el funcionamiento del acelerómetro.



F

Figura 1.8 Diagrama de bloques del acelerómetro ADXL330

La parte del chip que detecta los movimientos es el sensor, formado por una pieza de silicio sujetada en uno de sus extremos (ver Figura 1.9), y colocado entre el campo eléctrico que generan dos capacitores. Se encarga de la transmisión del movimiento.

Cuando sometemos el mando a un desplazamiento, la pieza de silicio se desplaza hacia uno de los dos lados (capacitores), con mayor o menor deformación, provocando un cambio en el campo eléctrico; es decir creando una diferencia de potencial y variando su fase.

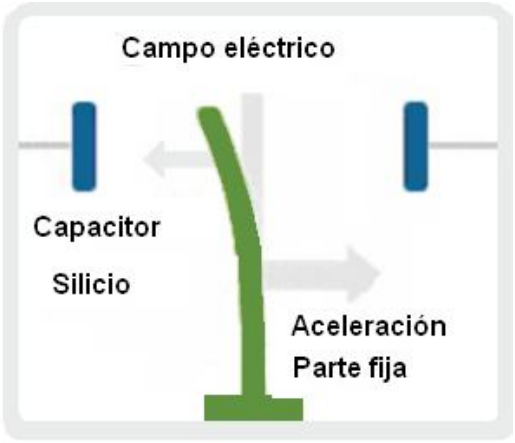


Figura 1.9 Esquema de funcionamiento de la pieza de silicio

En la parte interna se diferencian tres etapas:

- **Amplificador AC:**

  Este amplificador obtiene como entrada una diferencia de potencial (señal cuadrada modulada) y es el que se encarga de amplificarla y tratar de que esa amplitud sea lineal.

- **Demodulador:**

  La demodulación sensitiva a la fase se utiliza para determinar la magnitud y dirección de la aceleración.

- **Amplificador de salida:**

  La salida del demodulador es amplificada y llevada fuera del chip a través de un filtro pasabajo el cual se encargara de filtrar las señales alternas para así obtener una salida DC con un voltaje mínimo de 0.1[V] y un máximo de 2.8[V], este filtro nos ayudará a mejorar la resolución, eliminar el ruido y a prevenir el aliasing (es el efecto que vuelve indistinguibles a señales continuas distintas, cuando se las muestrea digitalmente).

1.3.5 Módulo Bluetooth

El mando del Wii incorpora un chip Bluetooth de la marca Broadcom, modelo BCM2042 para la comunicación entre consola y el propio Wii Remote.

Figura 1.10 Módulo Bluetooth BCM2042

La función de este chip es la comunicación con otros dispositivos Bluetooth, en cualquiera de los sentidos; es decir, del dispositivo Bluetooth del Wii Remote hacia cualquier otro dispositivo y a la inversa.

Se puede aprovechar dicha tecnología para la comunicación con un PC, por lo que debemos conocer el funcionamiento que tiene el protocolo Bluetooth (HID), ya que es por medio del cual el dispositivo proporciona el servicio de entrada y salida de datos.

Figura 1.11 Comunicación Bluetooth de Wii Remote con otros dispositivos Bluetooth

1.3.6 El vibrador (rumble)

El modelo del motor es el SEM 8728DA, con un voltaje de 3,3 [V_{DC}] y una corriente máxima de 35 [mA.], conformado con un pequeño motor conectado a un peso excéntrico. Al girar el motor esta excentricidad provoca la vibración.

Figura 1.12 Vibrador SEM 8728DA

El vibrador se activa cuando el bit correspondiente al vibrador se encuentra en 1, en caso contrario, estará desactivado.

1.4 Tecnología Bluetooth

1.4.1 Qué es Bluetooth

Bluetooth es un protocolo global de comunicación inalámbrica establecido por la IEEE 802.15.1, donde se pueden realizar conexiones de Red Inalámbricas teniendo la posibilidad de transmitir voz, datos, imagen multimedia entre diferentes dispositivos utilizando la tecnología de radio frecuencia de corto alcance (2.4 GHz de frecuencia).

Figura 1.13 Aplicación de la comunicación Bluetooth

Su objetivo es el de simplificar las comunicaciones entre dispositivos informáticos, como ordenadores móviles, teléfonos móviles y otros dispositivos de mano, siendo totalmente compatibles los dispositivos de una clase con los de las otras.

También pretende simplificar la sincronización de datos entre los dispositivos y otros ordenadores.

1.4.2 Funcionamiento

Podemos distinguir dos categorías de Bluetooth bastantes populares:

- La clase 1,
- La clase 2

Esto refiriéndose a la potencia de transmisión.

Clase 1: La clase 1 es menos común, pero no es difícil encontrar algunos dispositivos que los usan. Su alcance mayor es de unos 100 metros de distancia.

Clase2: La clase 2 es el estándar más común y barato. Su alcance mayor es de unos 10 metros de distancia.

Entonces Bluetooth definiría un canal de comunicación de máximo 720 [Kb/s] con rango óptimo de 10 metros (opcionalmente 100 metros con repetidores).

Su frecuencia de tráfico, con la que trabaja, se encuentra en el rango de 2,4 a 2,48 [GHz] con amplio espectro y saltos de frecuencia con posibilidad de transmitir en Full Dúplex (comunicación bidireccional) con un máximo de 1600 saltos/s, los cuales se dan entre un total de 79 frecuencias con intervalos de 1[MHz].

Por todo, la potencia de salida para transmitir a una distancia máxima de 10 metros es de 0 [dBm] (1 [mW]), mientras que, en sí, la versión de largo alcance transmite entre los 20 [dBm] y 30 [dBm] (entre 100 [mW] y 1 [W]).

El dispositivo Bluetooth se compone fundamentalmente, de dos partes muy importantes: en primer lugar, un dispositivo de radio (encargado de transmitir y modular la señal), y el controlador digital (compuesto por un procesador de señales digitales, un CPU y de los diferentes interfaces con el dispositivo anfitrión.

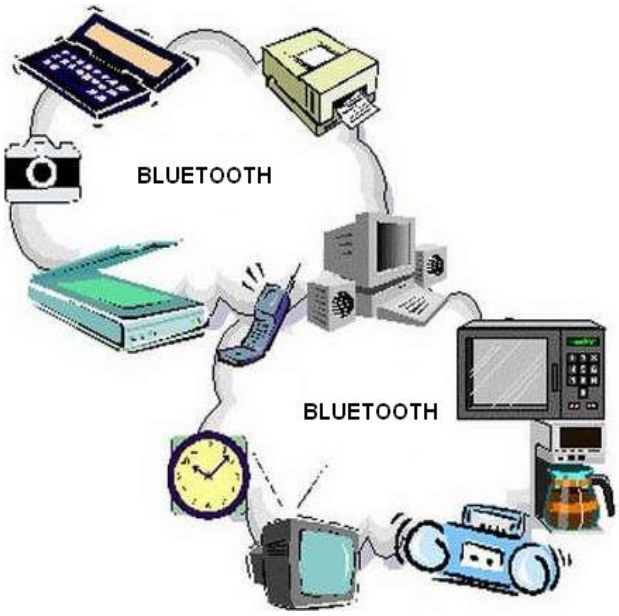


Figura 1.14 Diversas formas de comunicación Bluetooth

Se define anfitrión, como aquel dispositivo que utiliza o solicita los servicios de un dispositivo de interfaz humana (HID).

1.4.3 Ventajas

Una de las ventajas que encontramos con la tecnología de Bluetooth es que al momento de comunicar los dispositivos podemos hacerlo en un corto o largo alcance, de forma sencilla y cómoda, y sin la necesidad de estar lidiando con los tipos de cables.

Otra de las ventajas que tiene es que está disponible en una amplia variedad de dispositivos, desde teléfonos móviles hasta instrumentos médicos, automóviles, redes inalámbricas, etc.

También sirve para crear una conexión a Internet inalámbrica desde tu portátil usando tu teléfono móvil. Un caso aún más práctico es el poder sincronizar libretas de direcciones, calendarios, etc. en tu PDA, teléfono móvil, ordenador de sobremesa y portátil automáticamente y al mismo tiempo.

Bluetooth es mejor para la difusión de información, incluso a través de obstáculos.

1.5 Software LabVIEW

1.5.1 Definición

LabVIEW es un revolucionario ambiente de desarrollo gráfico con funciones integradas para realizar adquisición de datos, control de instrumentos, análisis de mediciones y presentaciones de dato. LabVIEW

da flexibilidad de un poderoso ambiente de programación sin la complejidad de los ambientes tradicionales.

El lenguaje que utiliza es el lenguaje G.

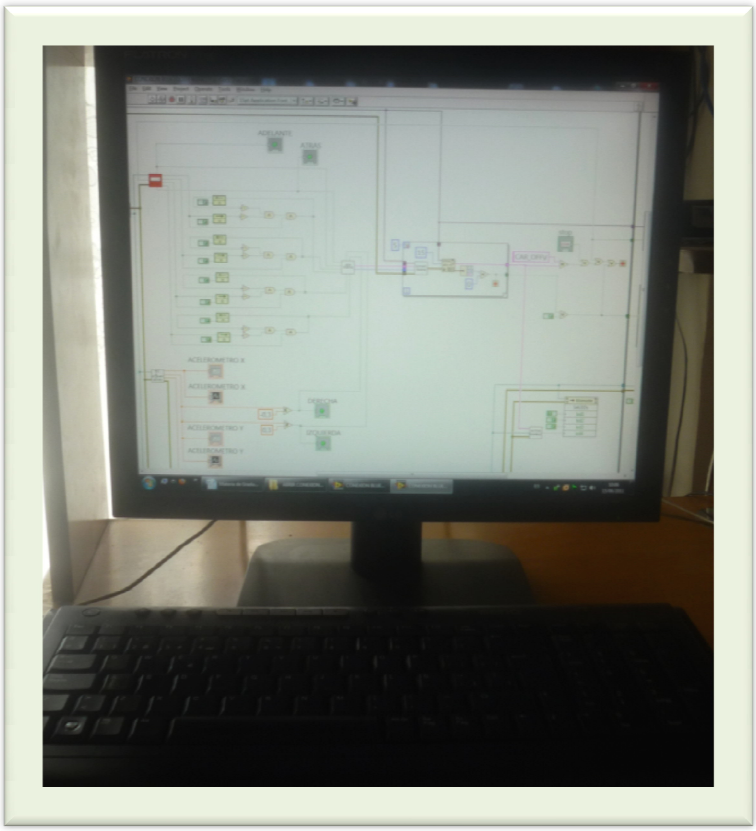


Figura 1.15 Entorno Gráfico

Los programas desarrollados con LabVIEW tienen extensión **.**vi (virtual instrument); esto nos indica que es un programa para manejar instrumentos.

1.5.2 Funcionamiento del software LabVIEW

Los programas desarrollados mediante LabVIEW se denominan instrumentos virtuales (VIs), porque su apariencia y funcionamiento imitan los de un instrumento real. Sin embargo son análogos a las funciones creadas con los lenguajes de programación convencionales. Los VIs tienen una parte interactiva con el usuario y otra parte de código fuente, y aceptan parámetros procedentes de otros VIs.

Todos los VIs tienen un panel frontal y un diagrama de bloques, también podemos hacer uso de las paletas ya que ellas contienen las opciones que se emplean para crear y modificar los VIs.

Con el entorno gráfico de programación de LabVIEW se comienza a programar a partir de un Panel Frontal y luego se finalizará con el diagrama de bloques.

A continuación se definirán el Panel Frontal y el Diagrama de Bloques.

1.5.2.1 Panel Frontal

Se trata de la interfaz gráfica del VI con el usuario. Esta interfaz recoge las entradas procedentes del usuario y representa las salidas proporcionadas por el programa.

Un panel frontal está formado por una serie de botones, pulsadores, potenciómetros, gráficos, etc. Cada uno de ellos puede estar definido como:

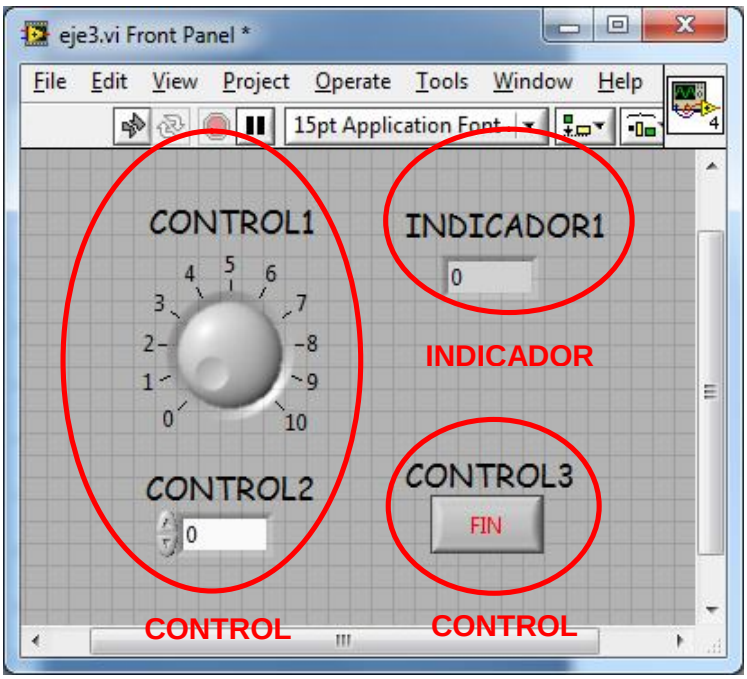


Figura 1.16 Panel de Control de LabVIEW

- **Un control:**

  Los controles sirven para introducir parámetros al VI.

- **Un indicador:**

  Los indicadores se emplean para mostrar los resultados producidos, ya sean datos adquiridos o resultados de alguna operación.

1.5.2.2 Diagrama de Bloques

El diagrama de bloques constituye el código fuente del VI. El diagrama de bloque es donde se realiza la implementación del programa del VI para controlar o realizar cualquier procesamiento de las entradas y salidas que se crearon en el panel frontal.

El diagrama de bloques incluye funciones y estructuras integradas en las librerías que incorpora LabVIEW. En el lenguaje G las funciones y las estructuras son nodos elementales.

Los controles e indicadores que se colocaron previamente en el Panel Frontal, se materializan en el diagrama de bloques mediante los terminales.

El diagrama de bloques se construye conectando los distintos objetos entre si, como si se tratara de un circuito, los cables se encargarán de unir los terminales de entrada y salida con sus correspondientes objetos, ya que los cables son los que van a facilitar el flujo de datos desde su origen hasta su destino, ya sea una función, una estructura, un terminal, etc.

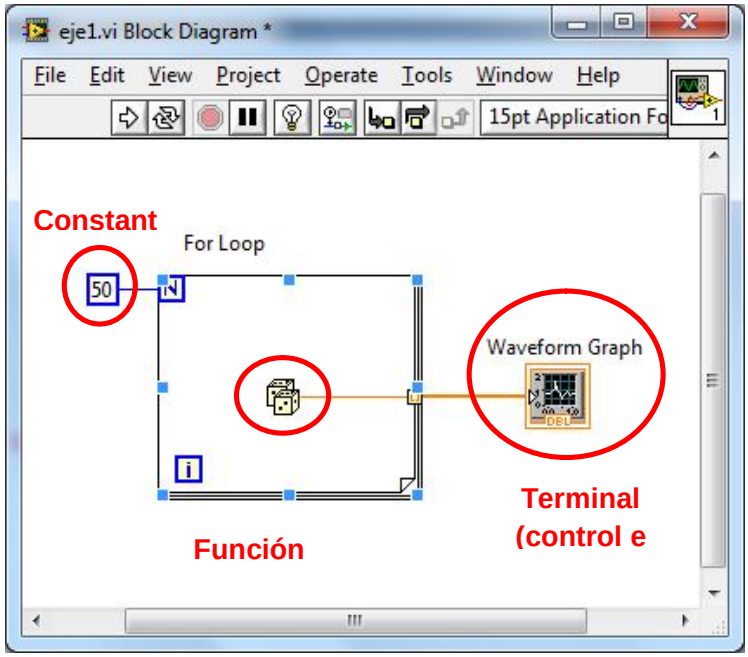


Figura 1.17 Diagrama de bloques de LabVIEW

Cada cable tiene un color o un estilo diferente y esto es lo que nos permite diferenciar unos tipos de datos de otros.

1.5.3 Aplicaciones

LabVIEW se ha convertido en una herramienta de desarrollo estándar de la industria para aplicaciones de prueba, esto se debe a que LabVIEW en combinación con el entorno de ejecutor de pruebas (Test Stand), de National instrument y la librería de controladores de instrumentos proporciona una plataforma de pruebas consistente e integrada para un sistema completo.

LabVIEW es usado principalmente por ingenieros y científicos para tareas como:

- Adquisición de datos.
- Control de instrumentos.
- Automatización industrial.
- Diseño de control: prototipos rápidos y hardware en el bucle (HIL).

LabVIEW a su vez nos facilita las realizaciones de los siguientes puntos que a continuación se detallan:

- **Lenguaje Desarrollado para Medida, Control y Automatización:**
  A diferencia de los lenguajes de propósito general, LabVIEW tiene funciones específicas para acelerar el desarrollo de aplicaciones de medida, control y automatización.

- **Entorno de Desarrollo Intuitivo para aumentar la Productividad**:
  LabVIEW proporciona herramientas muy potentes para crear aplicaciones sin líneas de código. Con LabVIEW podemos colocar objetos ya construidos para crear interfaces de usuario rápidamente. Y finalmente lo que nos tocaría realizar es la especificación de las funciones del sistema construyendo los diagramas de bloques respectivos.

- **Fácil Integración con Miles de Instrumentos y Dispositivos de Medida:**
  LabVIEW puede conectarse de manera transparente con todo tipo de hardware incluyendo instrumentos de escritorio, tarjetas insertables, controladores de movimiento y controladores lógicos programables (PLCs).

- **Entorno Abierto para Usar con Otras Aplicaciones**: Con LabVIEW podemos conectarnos a otras aplicaciones y compartir datos a través de:
  - ActiveX,
  - Web,
  - DLLs,
  - Librerías compartidas,
  - SQL,
  - TCP/IP,
  - XML,
  - OPC y otros.

- **Optimizar el desarrollo del Sistema:**

  En muchas aplicaciones, la velocidad de ejecución es vital y ya que LabVIEW cuenta con un compilador incluido este nos genera un código mas optimizado.

  Con LabVIEW podemos desarrollar sistemas que cumplan con los diversos requisitos de desarrollo a través de las plataformas incluyendo Windows, Macintosh, UNIX y sistemas de tiempo real.

1.5.4 Ventajas del Software LabVIEW

Las ventajas que proporcionan el empleo de LabVIEW se resumen en las siguientes:

- Se reduce el tiempo de desarrollo de las aplicaciones al menos de 4 a 10 veces, ya que es muy intuitivo y fácil de aprender.
- Dota de gran flexibilidad al sistema, permitiendo cambios y actualizaciones tanto del hardware como del software.
- Da la posibilidad a los usuarios de crear soluciones completas y complejas.
- Con un único sistema de desarrollo se integran las funciones de adquisición, análisis y presentación de datos.
- El sistema está dotado de un compilador gráfico para lograr la máxima velocidad de ejecución posible.
- Tiene la posibilidad de incorporar aplicaciones escritas en otros lenguajes.

CAPITULO 2

Descripción general del proyecto “Wii Remote with LabVIEW”

2.1 Diagrama de Bloques

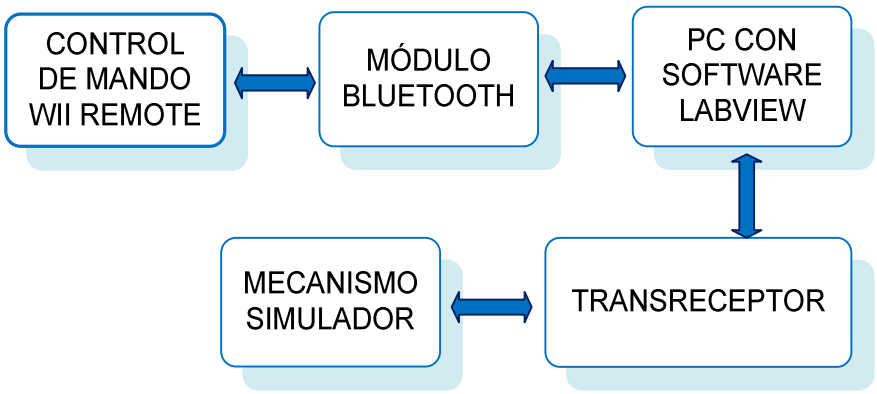


Figura 2.1 Diagrama de bloques

2.2 Descripción del diagrama de bloques

2.2.1 Control de mando Wii Remote

El mando Wii Remote se encarga de enviar en una trama de datos la información tal como el nombre, la identificación, dirección MAC, etc.,

cuya longitud es configurable dependiendo de las necesidades del usuario.

A esta trama se la llama reporte y, dependiendo del número de reporte, esta trama tendrá cierta información como por ejemplo, el reporte 0x20 es “Reporte de Estado” el cual envía en una trama de datos los botones presionados, el estado de cada led indicador de jugador, el nivel de las baterías y si están a punto de descargarse las baterías.

Además los reportes pueden ser de 2 tipos:

1. Entrada: son aquellos reportes que envía el Wii Remote hacia el terminal.

2. Salida: son aquellos reportes que envía el terminal hacia el Wii Remote.

Siguiendo con el ejemplo anterior el reporte de estado es un reporte de entrada, dado que envía información del Wii Remote para conocimiento del terminal.

2.2.2 Módulo Bluetooth

Este Módulo Bluetooth utiliza una arquitectura de protocolos que permiten el intercambio transparente de información entre aplicaciones diseñadas de acuerdo con dicha especificación y fomentan la interoperabilidad entre los productos de diferentes fabricantes.

En este caso el Módulo Bluetooth empieza enviando una señal al aire y espera quien le envíe una respuesta para establecer con él un camino (entre la PC y el Bluetooth), al recibir esta respuesta estos dispositivos quedan emparejados.

En nuestro proyecto el protocolo Bluetooth utilizado ya viene incorporado en la PC, a este protocolo se lo conoce como Toshiba Bluetooth Stack.

2.2.3 PC con software LabVIEW

Una vez que están emparejados los dispositivos hacemos el uso del Software LabVIEW para la interconexión con el mando Wii Remote.

El Wii Remote siempre va a estar enviando reportes de estado al software

LabVIEW para que este se encargue de procesarlas, el mismo donde podemos visualizar y representar los datos por medio de la PC para así enviar la información requerida por el usuario hacia el mecanismo teleoperado.

LabVIEW envía una trama de 24 bits, de los cuales los 8 bits más significativos serán de las operaciones que LabVIEW desea que realice el mecanismo teleoperado los siguientes 8 bits son para la detección de errores, utilizando un código de redundancia cíclica con polinomio de grado 4, siguiendo la norma ITU-T G-704 el polinomio generador es el siguiente: x^4+x+1 y los 8 menos significativos son para el fin de trama.

2.2.4 Transreceptor

Para este bloque se usó el transreceptor HM-TR de la compañía HopeRF, y se lo usa para poder enviar los datos desde el LabVIEW hacia el mecanismo teleoperado. Este módulo posee una modulación FSK en banda 433MHz, transmisión semiduplex, con frecuencia central de 433.15MHz, y está configurado con comunicación serial a 9600 bps.

Este módulo fue configurado para transmitir paquetes de 8 bits hacia el mecanismo teleoperado.

2.2.5 Mecanismo teleoperado

El mecanismo teleoperado fue comprado para luego modificar su circuitería electrónica.

Internamente se utilizó un transreceptor HM-TR, un microcontrolador PIC 16F628A de Microchip y 2 puentes h para el manejo de dirección de los motores.

El mecanismo teleoperado recibirá las órdenes de LabVIEW en forma de trama de 24 bits, el microcontrolador procederá a desenmascarar la información que se recibió y la ejecutará.

Así como LabVIEW, el controlador del mecanismo teleoperado debe tener el mismo polinomio generador del código de redundancia cíclica, para poder discernir si el dato que llegó está correcto o fue alterado por el ruido del medio.

Capítulo 3

Realización de prototipo

3.1 Panel Frontal

El panel frontal consta de dos partes, la primera nos muestra el estado del Wii Remote y la segunda el estado del mecanismo teleoperado.

En esta primera parte, se muestra el nivel de las baterías mediante un tanque ubicado en la paleta de control de LabVIEW, el porcentaje de carga de las mismas y, si las baterías necesitan cargarse.

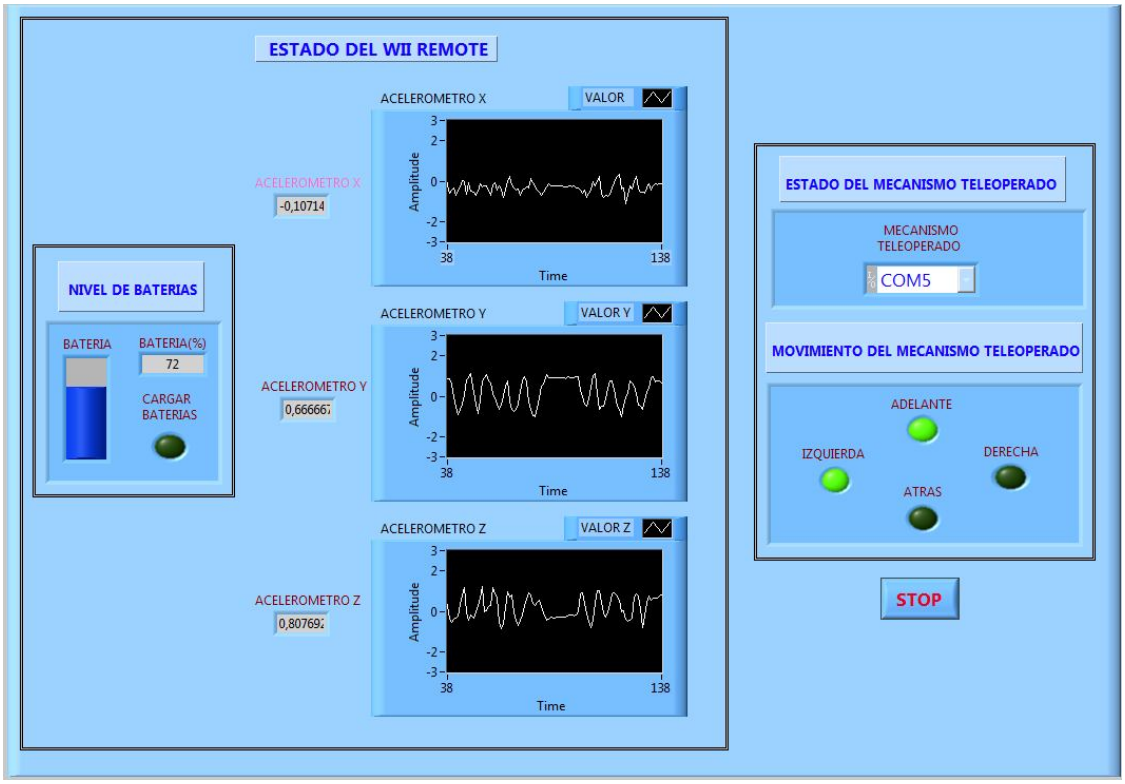


Figura 3.1 Panel Frontal

3.2 Diagrama de Bloques

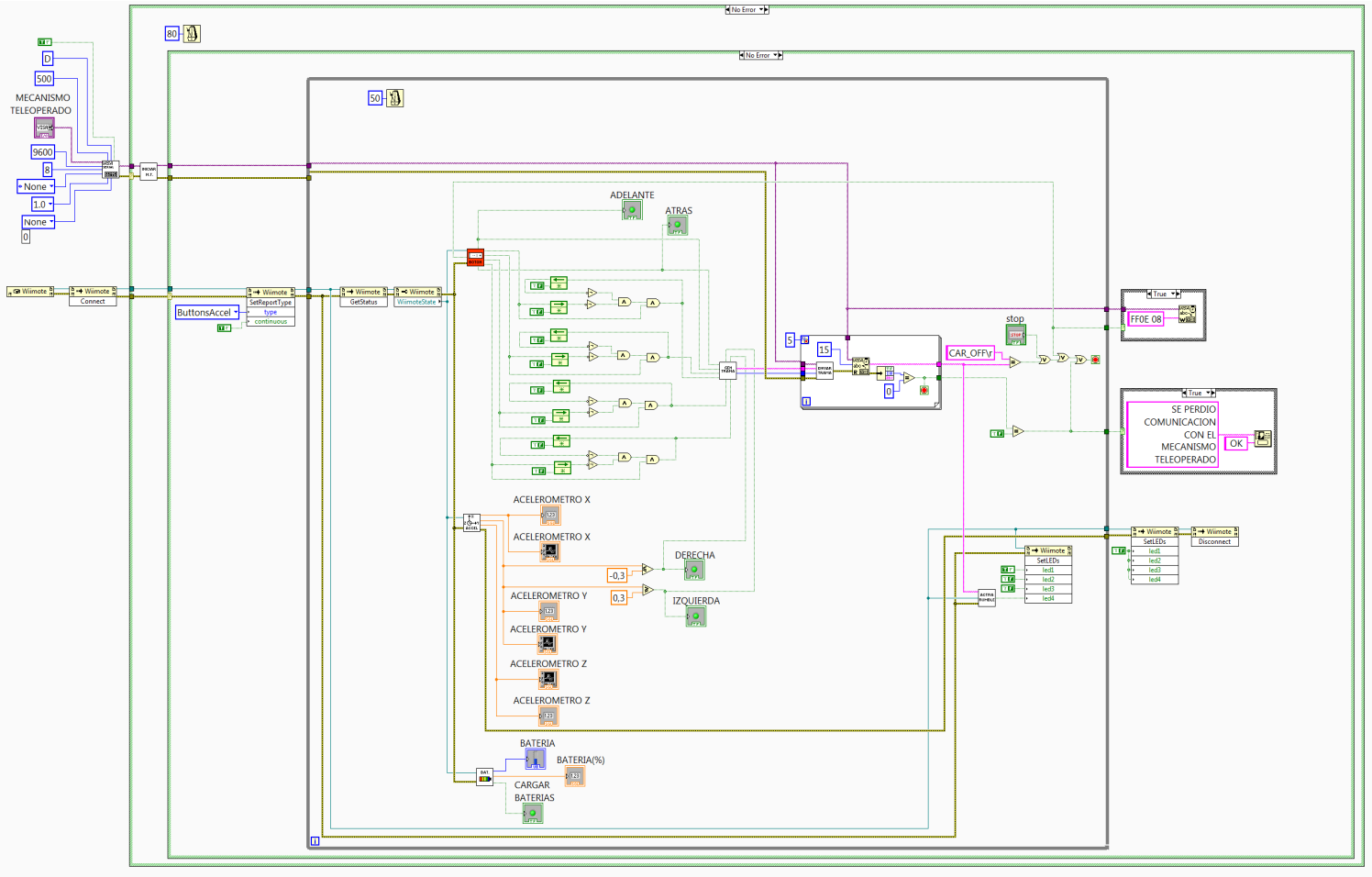


Figura 3.2 Diagrama de Bloques

3.3 Explicación del programa de LabVIEW

La primera etapa es la inicialización de comunicaciones. La figura 3.3 es el bloque de inicialización de comunicación con el mecanismo teleoperado, este canal se establece en 9600 bps, con 8 bits, sin bit de paridad y 1 bit de parada; luego de este bloque se encuentra un lazo CASE que revisa la señal de error de la inicialización del canal serial como se muestra en la figura 3.4.

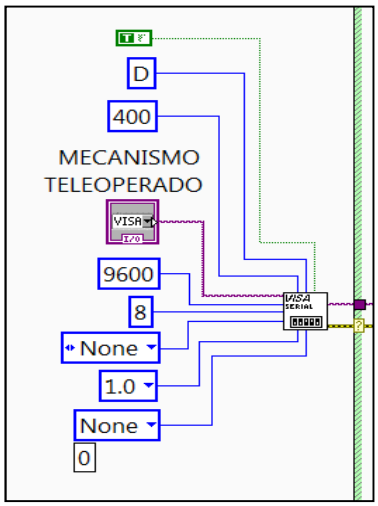


Figura 3.3 Inicialización de puerto serial

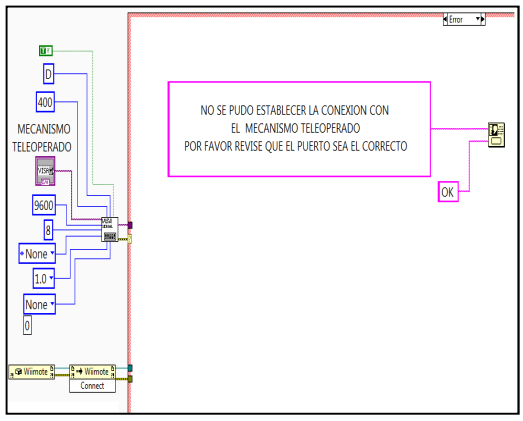


Figura 3.4 Lazo case, error de comunicación

Dentro de este lazo se encuentra otro lazo CASE y un SubVI llamado “INICIAR MECANISMO TELEOPERADO”. La figura 3.5 es un SubVI que posee el proceso para iniciar la transferencia de datos entre el LabVIEW y el mecanismo

teleoperado, si no se puede inicializar el puerto serial se envía una señal de error y entra al lazo CASE “Error“ que envía un mensaje de error indicando “NO SE PUDO ESTABLECER LA CONEXIÓN CON EL MECANISMO TELEOPERADO. POR FAVOR REVISE QUE EL PUERTO SEA EL CORRECTO”; en caso de que no ocurra un error, entra al lazo CASE “No Error”, donde se encuentra otro lazo CASE que revisa la señal de Error del estado de conexión del Wii Remote.

La Figura 3.6 es el contenido del SubVI “INICIAR MECANISMO TELEOPERADO”, como se observa LabVIEW envía al mecanismo teleoperado la trama “STATUS_OK\r”, luego del cual el mecanismo teleoperado enviará “CAR_ON\r” y se iniciará el funcionamiento del mismo; en caso que no se reciba “CAR_ON\n” se vuelve a enviar “STATUS_OK\r” y así consecutivamente.

Este SubVI devuelve la señal Error y el nombre de puerto serial que luego ingresará en un lazo While dentro del segundo lazo CASE.

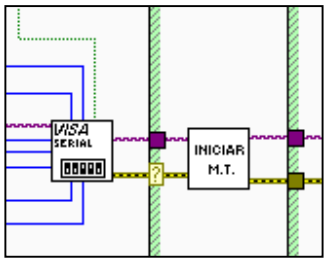


Figura 3.5 SubVI Iniciar Mecanismo Teleoperado

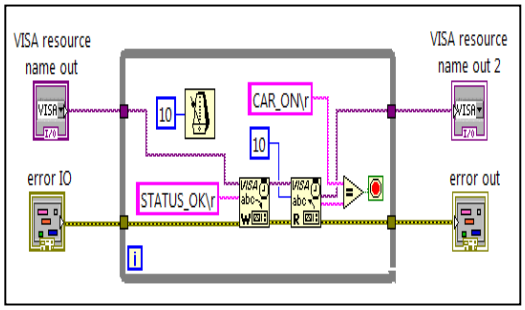


Figura 3.6 Composición interna del SubVI
Iniciar mecanismo teleoperado

La figura 3.7 muestra como se inicializa el módulo Bluetooth para la comunicación con el Wii Remote, para esto usaremos el Nodo Constructor y un nodo de invocación.

En el nodo constructor añadiremos la librería “WiimoteLib”; este nodo nos genera una referencia que contiene en su interior la información de la conexión y las funciones disponibles en la librería. Esta referencia se la usará en un nodo de invocación. En el nodo de invocación llamaremos a la función “connect”, la cual establecerá la conexión con el Wii Remote. La señal de error que envíe este nodo de invocación es la señal que se usa en el segundo lazo CASE; si la señal es “Error” se envía el mensaje “SU Wii REMOTE NO FUE ENCONTRADO. POR FAVOR ENLACE SU Wii REMOTE A LA RED BLUETOOTH. POR SEGURIDAD EL MECANISMO TELEOPERADO FUE

DESCONECTADO", tal como se muestra en la figura 3.8. Si la señal es "No Error" entra en el lazo que lee la trama que envía el Wii Remote, la procesa y envía las órdenes al mecanismo teleoperado.

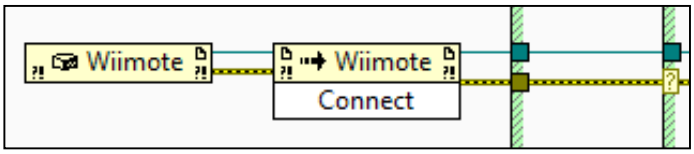


Figura 3.7 Inicialización y conexión con el Wii Remote

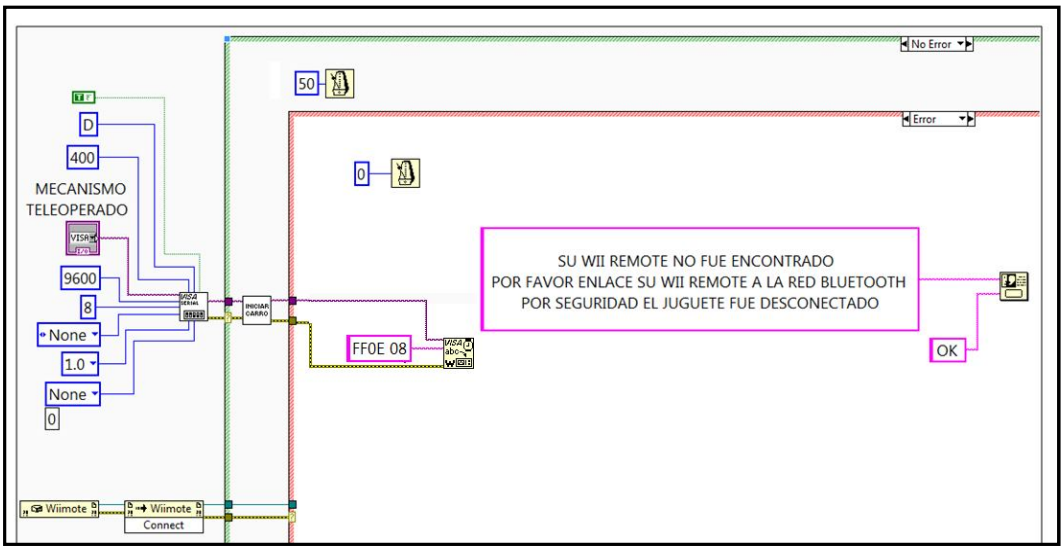


Figura 3.8 Lazo case error de enlace con el Wii Remote

Una vez que ingresa al segundo lazo CASE como indica la figura 3.9. La referencia se utiliza en un nuevo nodo de invocación, esta vez se usará la función "SetReportType", con reportes que serán enviados periódicamente; ya

sea por cambios de estado en los botones, acelerómetros, cámara, etc. Estos reportes contienen la información de los botones y acelerómetros.

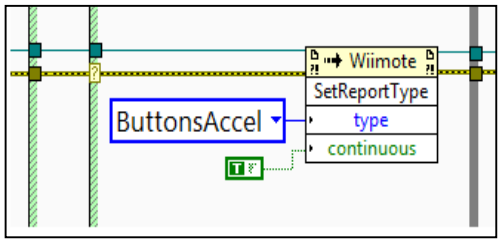


Figura 3.9 Configuración de reportes

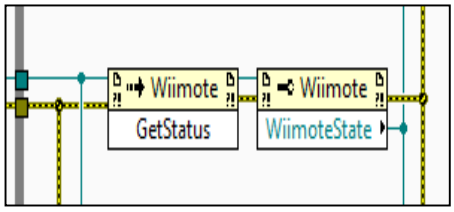


Figura 3.10 Obtención de reportes de estado del Wii Remote

Una vez hecho todo lo anterior, se ingresará en un lazo While que se repetirá cada 50ms. El primer bloque que encontramos es un nodo de invocación que llama la función “GetStatus”, para leer el reporte de estado del Wii Remote.

Luego de esto llamamos a la función “WiimoteState” (Figura 3.10); esta función devuelve una nueva referencia que contiene la información del reporte que se leyó. Posteriormente esta referencia se usará en 3 SubVI’s que procesarán la información de la referencia y extraerán la información.

El primer SubVI es BAT, que lee el nivel de las baterías del WiiRemote mediante la función “”BatteyRaw” (Figura 3.11) que devuelve un valor entero indicando el porcentaje del nivel de las baterías; esta información se muestra en el panel frontal mediante un indicador de tanque, en un indicador numérico se muestra el porcentaje del nivel de baterías y un indicador LED se encenderá si el nivel de las baterías es menor a 30%.

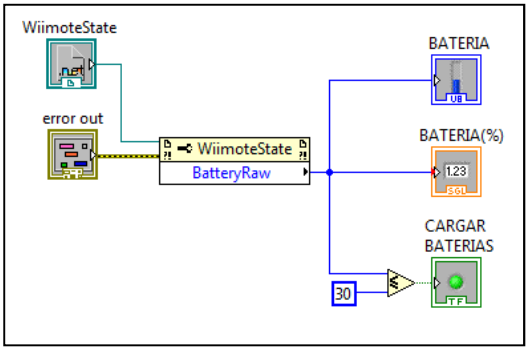


Figura 3.11 SubVI BAT

El Segundo SubVI es “ACCEL” (Figura 3.12), este Iniciar SubVI llama a la función “AccelState” la cual devuelve una referencia con la información de las funciones disponibles.

Ésta referencia se la usa en otro nodo de invocación y llamamos a la función “Values” que devuelve una referencia con los valores de los acelerómetros. Para leer los valores de los acelerómetros se usa un nodo de invocación, los valores leídos para el acelerómetro de **X, Y** y **Z** son de tipo punto flotante.

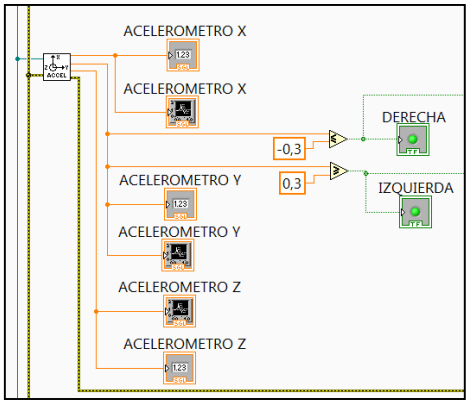


Figura 3.12 SubVI ACCEL

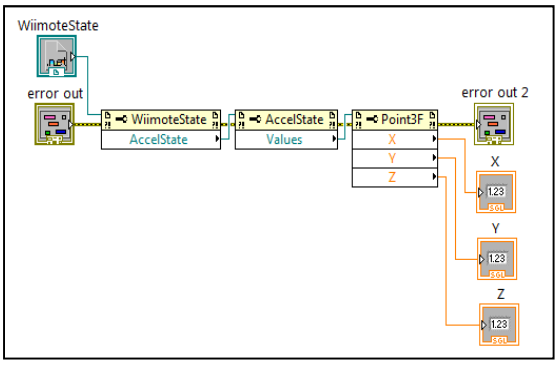


Figura 3.13 Indicadores gráficos conectados al SubVI ACCEL

Los valores leídos por el SubVI “ACCEL” se muestran en un Waveform Chart y en un indicador numérico (Figura 3.13). Para hacer uso del acelerómetro en la dirección del mecanismo teleoperado se usan comparaciones; si el valor medido en el acelerómetro Y es mayor a 0.3 [g] se toma como un giro a la izquierda, si es menor a -0.3 [g] se toma como un giro a la derecha.

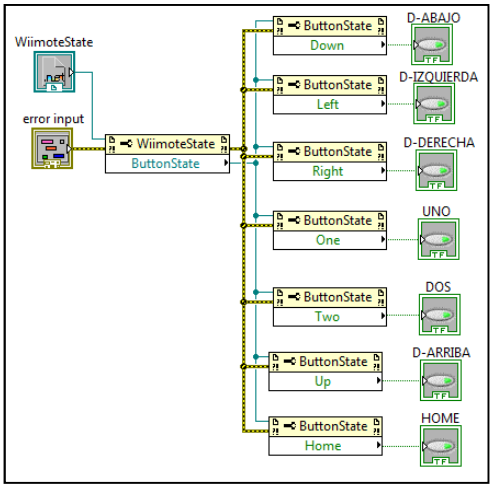


Figura 3.14 Composición interna del Iniciar Mecanismo teleoperado BOTÓN

El tercer SubVI es “BOTÓN”, internamente este SubVI llama a la función “ButtonState” (figura 3.14), esta función devuelve una referencia con la información de los botones.

Para acceder a la información requerida se utilizan nodos de invocación y se llama a la función del botón que necesitemos; estos nodos de invocación devuelven valores binarios (‘1’ si está presionado, ‘0’ si no está presionado).

Para este proyecto se utilizó el botón ‘1’ para aceleración del mecanismo teleoperado, el botón ‘2’ para retroceder, el D-Pad derecha para las luces delanteras, el D-Pad arriba para las luces izquierdas, el D-Pad izquierda para las luces traseras, el D-Pad abajo para las luces derechas.

Dado que las luces se activan con presionar algún botón de luces y se desactiva con volver a presionarlo, se implementó un bloque generador de Pulsos (Figura 3.15), que genera un pulso con duración de un ciclo del lazo while.

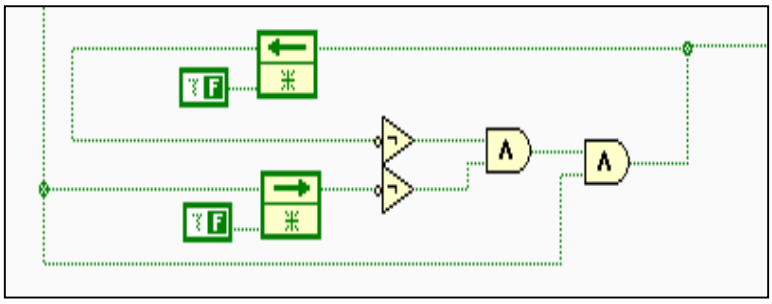


Figura 3.15 Generador de un pulso

Este bloque posee la operación lógica:

$$out = in \cdot \overline{in_{pasado}} \cdot \overline{out_{pasado}}$$

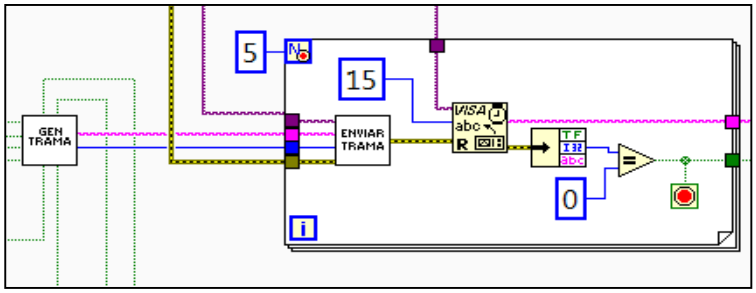


Figura 3.16 Bloque generador, transmisor y receptor de trama.

Cuando se ha obtenido el estado de todo lo necesario del Wii Remote, se genera la trama de datos mediante el SubVI “GEN TRAMA” (Figura 3.16), este SubVI genera la trama con el siguiente formato:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LUCES DELANTERAS	LUCES TRASERAS	DIRECCIONAL IZQUIERDA	DIRECCIONAL DERECHA	ADELANTE	ATRÁS	IZQUIERDA	DERECHA

Figura 3.17 Formato de la trama

Internamente el SubVI “GEN TRAMA” contiene un conjunto de selectores; es decir si un bit esta activado genera el código para ese bit como se puede observar en la figura 3.18.

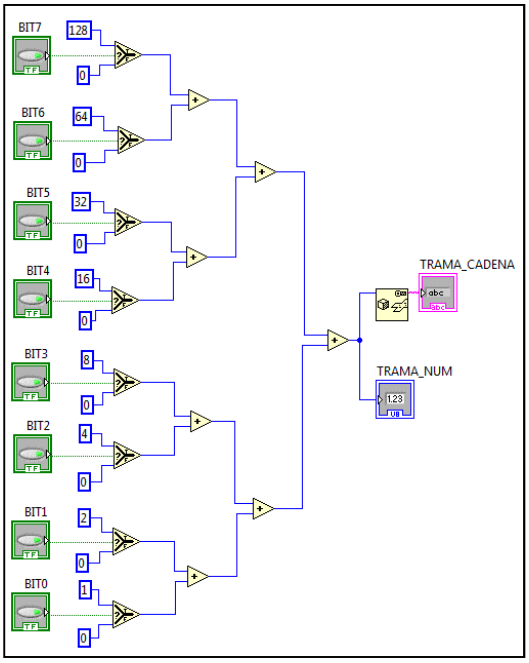


Figura 3.18 SubVI Gen Trama

Luego de este bloque se ingresa a un lazo FOR con 5 iteraciones, en las cuales se enviará la trama por el puerto serial hacia el mecanismo teleoperado; se usó un lazo FOR por el supuesto de que se pierda el mensaje en el canal; LabVIEW intentará transmitir su mensaje 5 veces hasta que responda el mecanismo teleoperado, ó se dará por terminada la comunicación.

Dentro de este lazo tenemos al SubVI "ENVIAR TRAMA", este SubVI toma la trama de datos generado, para generar el CRC4; luego se le añade el CRC4 a la trama generada y por último se inserta el fin de trama para luego enviarlo por

el puerto serial. Posterior al envío, se lee los datos del puerto serial; si se tarda mucho en leer datos significa que el mecanismo teleoperado no respondió al mensaje del LabVIEW, por lo que continúa el lazo FOR, ó si se terminaron las iteraciones se da por terminada la comunicación.

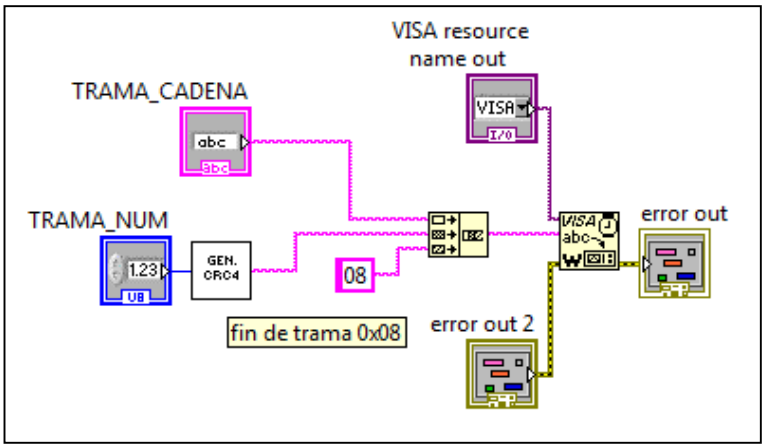


Figura 3.19 SubVI Enviar Trama

La última parte se encarga de revisar los datos leídos del puerto y terminar el programa de LabVIEW. La trama leída ingresa al SubVI “ACTIVA RUMBLE”, el cual se encargará de revisarla; esto es, si la trama es igual a 0x010D activa el Led4 y el rumble caso contrario no lo hace(Figura 3.20).

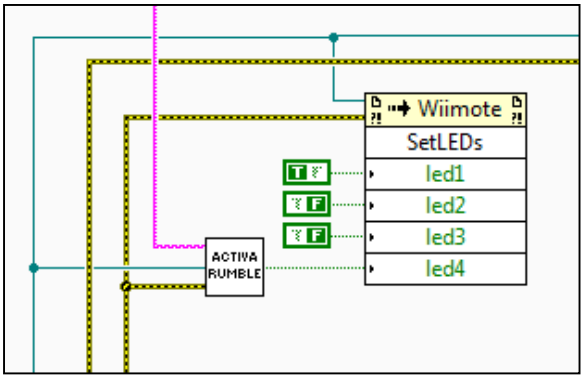


Figura 3.20 Bloque controlador del rumble

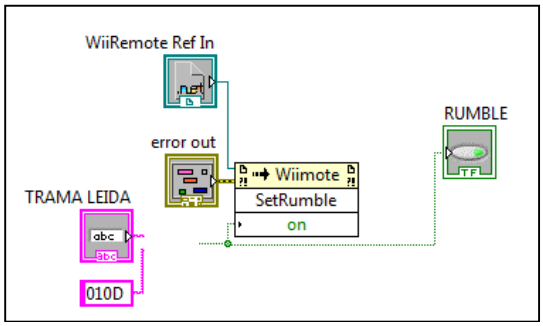


Figura 3.21 SubVI Activa Rumble

Como se dijo anteriormente el programa de LabVIEW se termina de 4 formas: la primera es cuando el mecanismo teloperado envía la trama “CAR_OFF\r” (Figura 3.22) indicando que el mecanismo teleoperado entró en proceso de apagado.

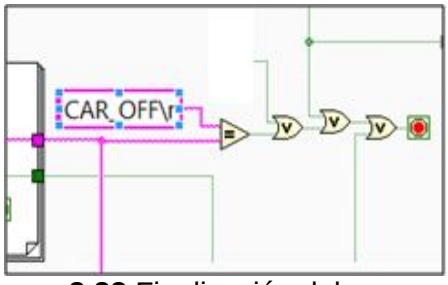


Figura 3.22 Finalización del programa desde el mecanismo teleoperado

La segunda manera es, si en el Wii Remote se presiona el botón Home, luego de esto se enviará un mensaje al mecanismo teleoperado para que termine la comunicación como se observa en la figura 3.23.

La tercera manera es cuando el LabVIEW no reciba ninguna respuesta del mecanismo teleoperado, luego del cual mostrará en pantalla un mensaje diciendo ” SE PERDIÓ COMUNICACION CON EL MECANISMO TELEOPERADO” como se observa en la figura 3.24.

La cuarta manera es con el botón STOP propio de LabVIEW(figura 3.25).

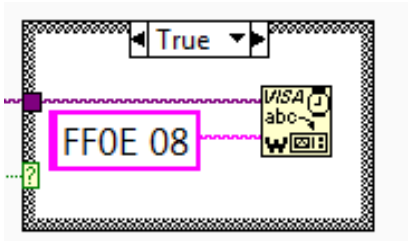


Figura 3.23 Envío de mensaje para terminar la comunicación con el mecanismo teleoperado

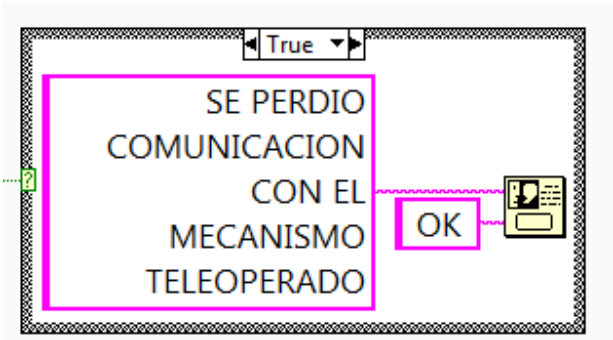


Figura 3.24 Presentación de mensaje al usuario

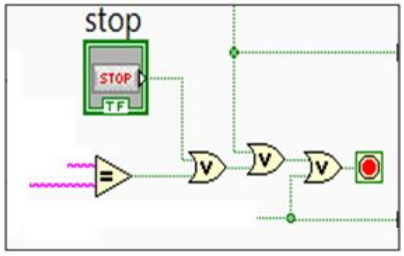


Figura 3.25 Finalización del programa desde el botón stop de LabVIEW

Cuando se llega a terminar el lazo WHILE, se procede a apagar los Leds indicadores del Wii Remote y posteriormente se terminará la comunicación con el mismo, como se muestra en la figura 3.26.

El motivo de apagar los Leds, es, que al no recibir ninguna trama el Wii Remote, éste muestra los últimos datos, por ejemplo si durante la activación del vibrador se pierde la comunicación con el Wii Remote, el vibrador permanecerá activado hasta que se lo vuelva a inicializar.

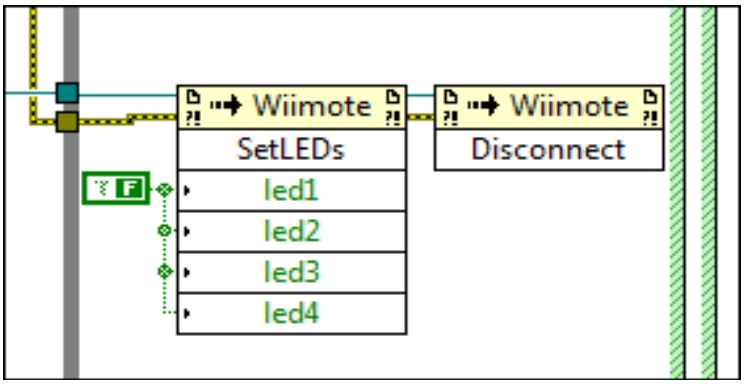


Figura 3.26 Desconexión del Wii Remote

Capítulo 4

Pruebas realizadas

Mecanismo teleoperado

Inicialmente se procede a colocar los cables en la parte inferior del mecanismo teleoperado que se unirán a la placas respectivas del mismo, como podemos observar en la figura 4.1.

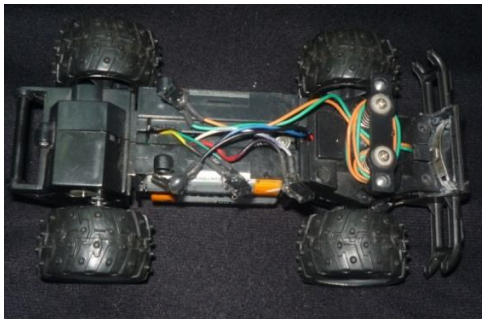


FIGURA 4.1 Estructura inferior del mecanismo teleoperado

Estas placas van colocadas en la parte superior del mecanismo teleoperado (figura 4.2)

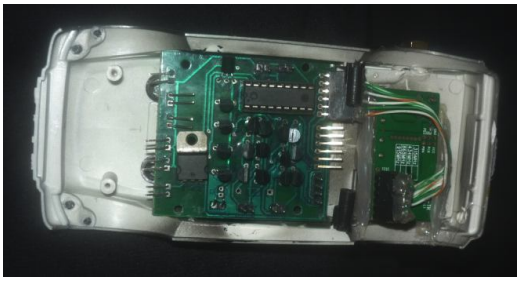


FIGURA 4.2 Estructura superior interna del mecanismo teleoperado

FIGURA 4.3 Estructura superior del mecanismo teleoperado

FIGURA 4.4 Estructura superior del mecanismo teleoperado, cara lateral con módulo HMTR

FIGURA 4.5 Estructura superior del mecanismo teleoperado, cara superior con Módulo HMTR

Luego de armar el mecanismo teleoperado se empieza primero por presionar el botón de comunicación con LabVIEW. Este botón indica al software “ESTOY LISTO”

Una vez que están listos se procede a probar cada una de sus funciones tales como:

- Movimientos hacia adelante.
- Movimiento hacia atrás.

- Giro hacia la izquierda.
- Giro hacia la derecha.
- Luces delanteras
- Luces traseras
- Direccional izquierdo
- Direccional derecho

FIGURA 4.6 Encendido de luces traseras

FIGURA 4.7 Encendido de direccional izquierdo

FIGURA 4.8 Encendido de luces delanteras

Transreceptor

FIGURA 4.9 Convertidor de USB a serial RS232

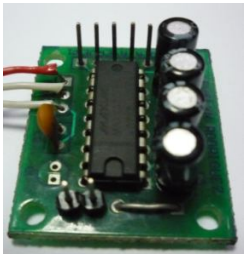


FIGURA 4.10 Convertidor de serial RS232 a Serial TTL

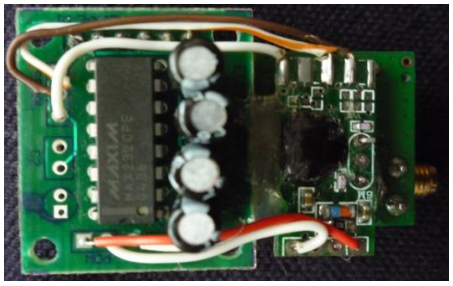


FIGURA 4.11 Placas acopladas al HMTR parte superior

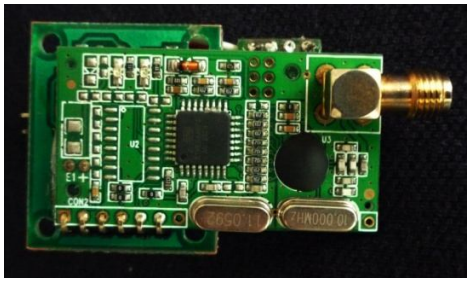


FIGURA 4.12 HMTR parte inferior

FIGURA 4.13 Disposición interna de los módulos

FIGURA 4.14 Acabado final del transreceptor

A continuación en el diagrama de bloques de LabVIEW se muestra la dirección que toma el mecanismo teleoperado.

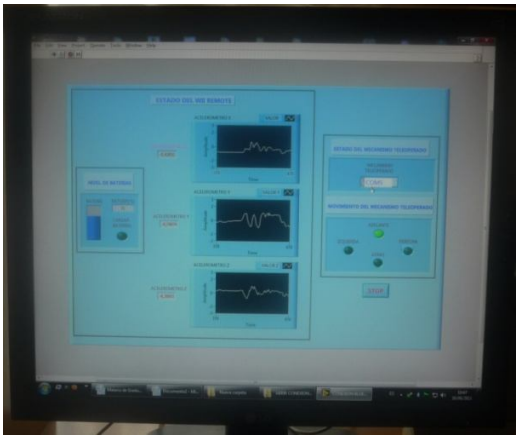


FIGURA 4.15 Mecanismo teleoperado dirigido hacia adelante

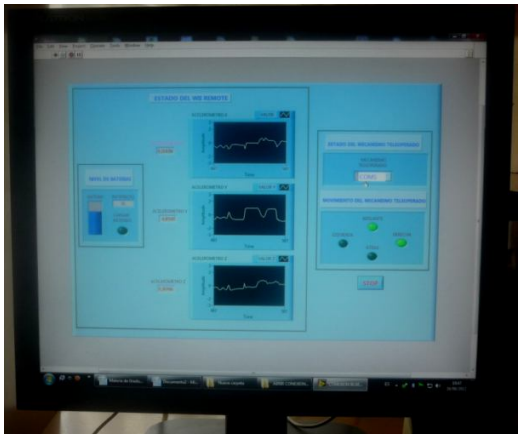


FIGURA 4.17 Mecanismo teleoperado moviéndose hacia adelante mientras gira a la derecha

FIGURA 4.16 Mecanismo teleoperado moviéndose hacia adelante mientras gira a la izquierda

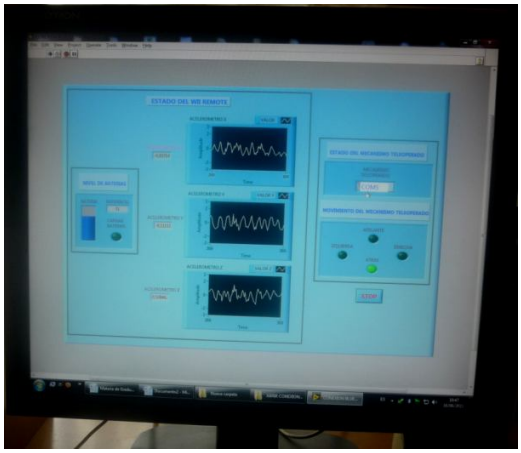


FIGURA 4.18 Mecanismo teleoperado en reversa

FIGURA 4.19 Muestra sólo el movimiento de las llantas a la izquierda

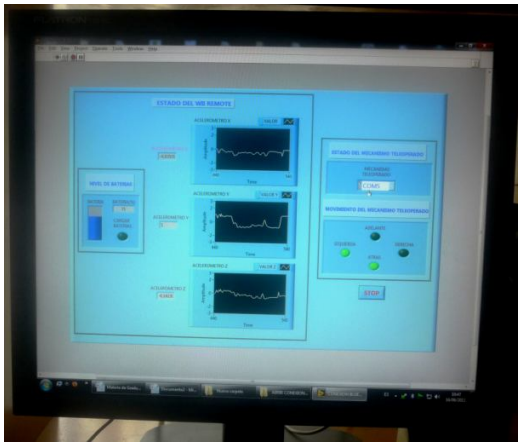


FIGURA 4.20 Mecanismo teleoperado moviéndose hacia atrás mientras gira a la izquierda

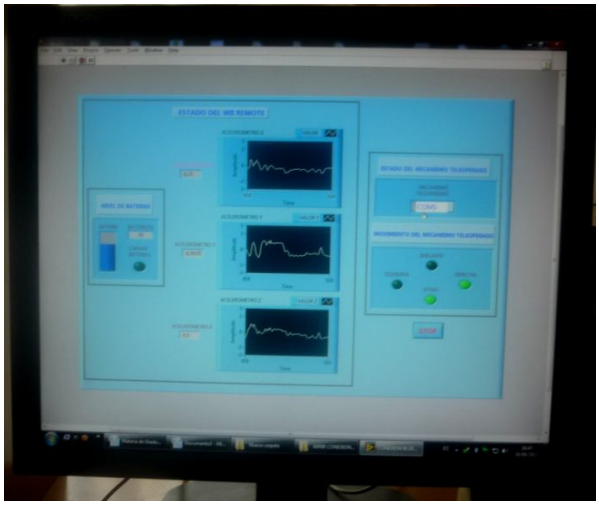


FIGURA 4.21 Mecanismo teleoperado dirigido hacia atrás mientras gira a la derecha

Conclusiones

1. Gracias al acelerómetro incluido en el Wii Remote, el manejo de cualquier estructura se realiza de forma más intuitiva, no obstante, y ante las necesidades de la aplicación desarrollada, fue necesario incluir unos umbrales de movimiento en uno de los ejes.

2. Es muy interesante la utilización del estándar Bluetooth HID, ya que la mayoría de sistemas operativos que podemos encontrar en el mercado reconocen perfectamente dispositivos USB HID, tales como teclados o ratones, sin la instalación de ningún driver especial.

3. En este proyecto hemos utilizado una herramienta de programación, como lo es LabVIEW. En particular, el uso de LabVIEW como interfaz de control persiste desde hace varios años, como consecuencia de la gran cantidad de librerías que contiene, que permiten la interacción con multitud de instrumentos y dispositivos electrónicos.

4. Para que el mecanismo teleoperado pueda discernir la información se agregó un microcontralador; PIC16F628A, que se va a encargar de enviar las señales de control para los bloques de potencia.

5. Unas de las aplicaciones que podemos dar a este proyecto es el manejo de un brazo robótico en vez del mecanismo teleoperado orientado a diversas áreas como la educación, la medicina, etc.

Recomendaciones

1. Se recomienda el uso de un módulo HMTR con modulación FSK porque tiene menos pérdida de señal (casi inmune al ruido), con una frecuencia de central de 433MHz ya que satisface la aplicación dada.

2. Debe tener en cuenta que la información que maneja el HMTR es de lógica serial TTL por lo que se debe implementar una forma de que los datos recibidos mediante USB sean entendidos por este.

3. Para una mayor duración del mecanismo teleoperado se recomienda el usó de 4 pilas alcalinas para la parte de potencia y 1 batería de 9V para la parte de control.

4. Debido que existen 3 ejes de trabajo en el Wii Remote, se recomienda para mayor comodidad del usuario, utilizar un solo eje, en este caso el eje Y para el manejo del mecanismo teleoperado que se encuentra en el rango de -1 y 1, lo que no ocurre con los otros dos ejes que se encuentran en el rango de 0 y 1 ó 0 y -1.

5. Se recomienda este proyecto para el posterior desarrollo de nuevas aplicaciones y proyectos.

A N E X O S

ANEXO A: Especificaciones del transreceptor HM-TR

HM-TR Series UHF Wireless Transparent Data Transceiver

General

The HM-TR series UHF wireless transparent data transceiver, developed by Hope Microelectronics Co. Ltd, is designed for applications that need wireless data transmission. It features high data rate, longer transmission distance, programmable frequencies, configurable UARTS formats and low sleep current make it ideal choice.

The communication protocol is self controlled and completely transparent to users. The module can be embedded to your existing design so that low cost high performance wireless data communication can be utilized easily.

Features

1. FSK (Frequency Shift Keying) modulation, high interference immunity.
2. 2-way half-duplex communication.
3. 315/433/868/915 MHz ISM band, globally license free.

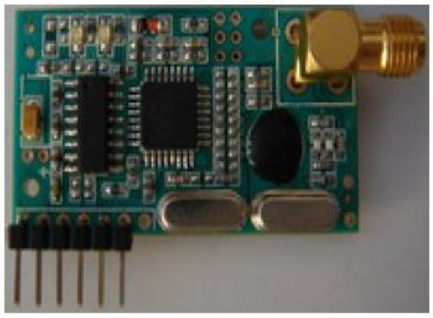
4. Programmable frequencies, allowing be used in FDMA(Frequency Division Multiple Accesses) applications.
5. Self controlled RF to UART protocol translation, reliable and easy to use.
6. Configurable UART format, with data rate from 300~19200bps.
7. Using ENABLE pin to control duty-cycle to satisfy different application requirements.
8. High performance, long transmission range, >300m in open area.
9. Standard UART interface, with TTL or RS232 logic level available.
10. Compact size, standard 0.1'' pinch SIP connector and SMA antenna socket.
11. No RF tuning needed in application.

Applications Areas

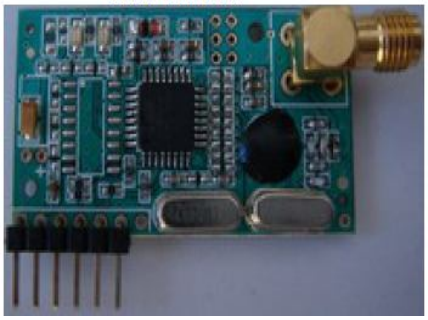
1. Remote control, remote measurement system.
2. Wireless metering.
3. Access control.
4. Identity discrimination.
5. Data collection.
6. IT home appliance.
7. Smart house products.
8. Data store and forwards repeater

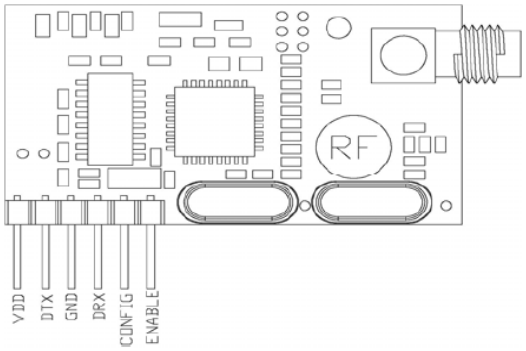
Overview and Pin assignment

HM-TR/232



HM-TR/TTL





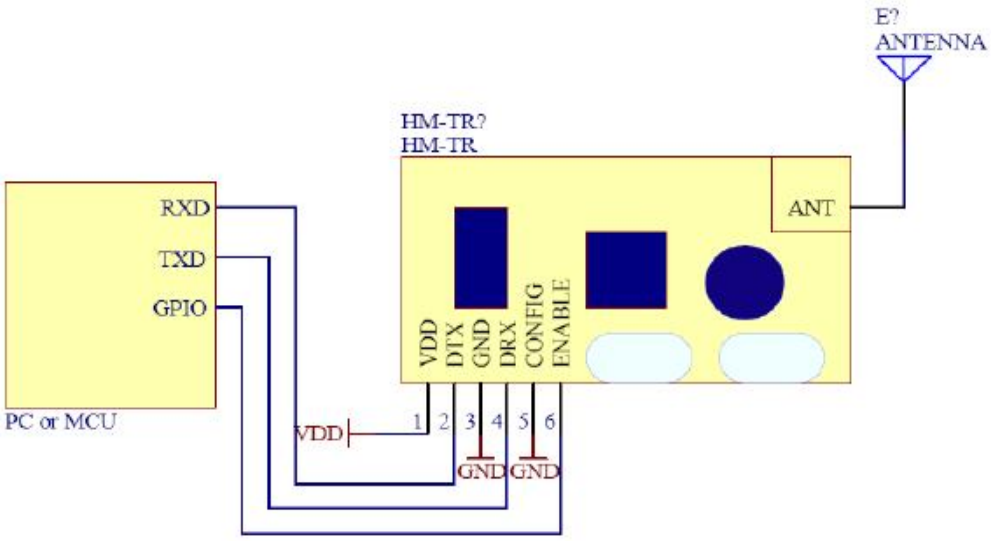
Pin	Name	Note
1	VDD	Power supply.
2	DTX	Data output from module.
3	GND	Ground.
4	DRX	Data input to module.
5	CONFIG	If this pin is high at power on, module will enter configure mode, while it communicates if set low.
6	ENABLE	If this pin is low in normal mode, the module will enter sleep mode immediately. Assert high will awaken.

Parameters

Parameter	Condition	Min	Typical	Max	
Power supply		4,5	5	5	V
Operate temperature		-3,5	25	80	C
Operate frequency	HM-TR433	430,24	434	439,75	MHz
	HM-TR868	860,48	869	879,51	
	HM-TR915	900,72	915	929,27	
Max output power	HM-TR433	3	5		dBm
	HM-TR868	-2	0		
	HM-TR915	-2	0		
Transmititing power		Pmax-21	Pmax		dBm
Receive sensitivity	HM-TR433		-105	-100	dBm
	HM-TR868		-102	-95	
	HM-TR915		-102	-95	
TX current	HM-TR433			26	mA
	HM-TR868			28,5	
	HM-TR915			30	
RX current	HM-TR433			15	mA
	HM-TR868			16	
	HM-TR915			17	
Sleep current	HM-TR433/TTL			1	uA
	HM-TR868/TTL			1	
	HM-TR915/TTL			1	
Reference distance	HM-TR433/TTL			330	m
	HM-TR868/TTL			220	
	HM-TR915/TTL			230	
Modulate deviation		15		240	kHz
Receiver bandwidth		67		400	kHz
UART data rate		300	9600	19200	bps
UART data bits		5	8	9	bit
UART parity check		None	Odd	Even	
UART stop bits		1	1	2	bit
ANT connector					SMA female
Module size					24x43mm

Quick Setup

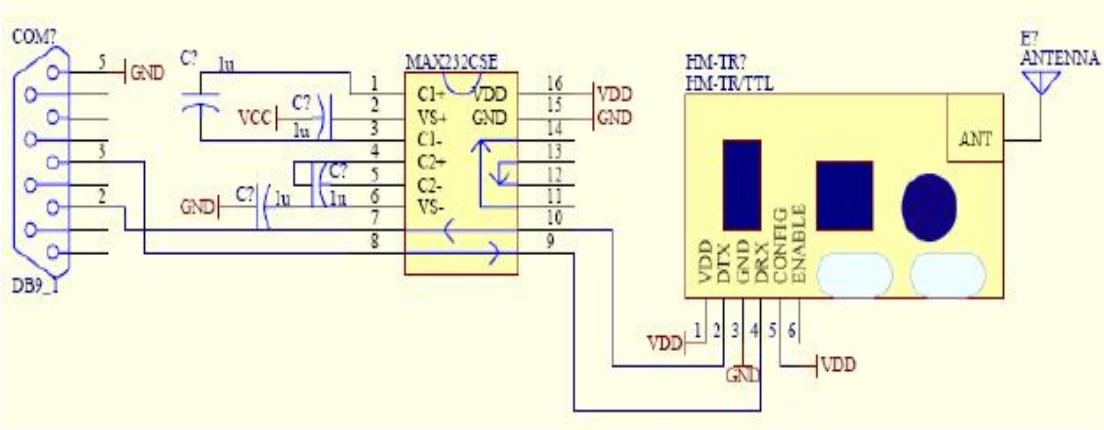
Connect HM-TR/232 to the Rs232 connector of serial fitted PC or connect HM-TR/TTL to MCU (micro controller unit)'s UART directly, apply power supply, both RED and Green status LED will blink 3 times to indicate it is ready for your application. If CONFIG pin is low at power on, module will enters normal mode for data transmission, or CONFIG is high the module enters configure mode to setup work parameters.



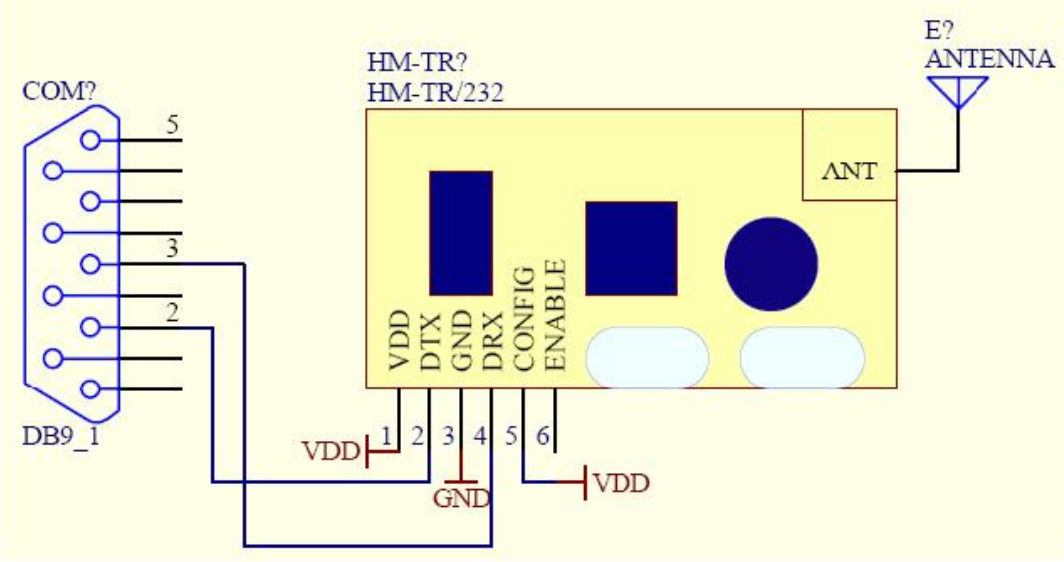
Note: MCU= Micro Controller Unit, PC=Personal Computer, GPIO=General Purpose Input/output

Normal mode connection

In normal mode, the ENABLE pin controls the module work or sleep, module will enter sleep mode as soon as the pin is low level.



Configure mode connection (HM-TR/TTL)



Configure mode connection (HM-TR/232)

ANEXO B: Especificaciones del PIC 16F628A

18–pin Flash-Based, 8-Bit CMOS Microcontrollers

With nano Watt Technology

High-Performance RISC CPU:

- Operating speeds from DC– 20MHz.
- Interrupt capability.
- 8-level deep hardware stack.
- Direct, Indirect and relative Addressing modes.
- 35 single-word instructions:
  - All instructions single cycle except branches.

Special Microcontroller Features:

- Internal and external oscillator options:
  - Precision internal 4MHz oscillator factory calibrated to +/- 1%.
  - Low-power internal 48 kHz oscillator.
  - External Oscillator support for crystals and resonators.
- Power-saving Sleep mode.
- Programmable weak pull-ups on PORTB.
- Multiplexed Master Clear/Input-pin.
- Watchdog Timer with independent oscillator for reliable operation.

- Low-voltage programming.
- In-Circuit Serial Programming™ (via two pins).
- Programmable code protection.
- Brown-out Reset.
- Power-on Reset.
- Power-up Timer and Oscillator Start-up Timer.
- Wide operating voltage range(2.0-5.5V).
- Industrial and extended temperature range.
- High-Endurance Flash/EEPROM cell:
  - 100,000 write Flash endurance.
  - 1,000,000 write EEPROM endurance.
  - 40 year data retention.

Low-Power Features:

- Standby Current:
  100 nA@2.0V,typical.
- Operating Current:
  - 12μA@32kHz,2.0V, typical.
  - 120 μA@1MHz,2.0V, typical.
- Watchdog Timer Current:
  - 1 μA@ 2.0V, typical.
- Timer1 Oscillator Current:
  - 1.2 μA@ 32kHz, 2.0V, typical.
- Dual-speed Internal Oscillator:
  - Run-time selectable between 4 MHz and 48kHz.
  - 4 μs wake-up from Sllep, 3.0V, typical.

Peripheral Features:

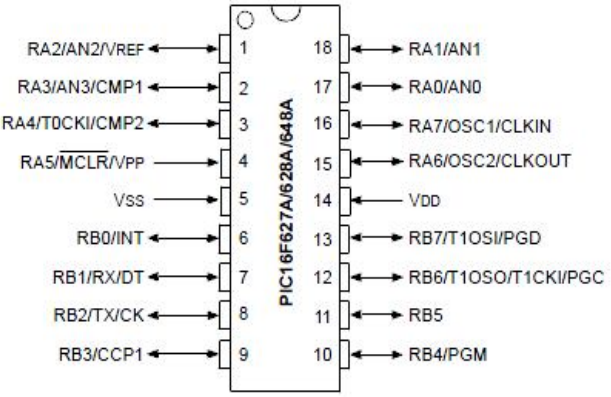
- 16 I/O pins with individual direction control.
- High current sink/source for direct LED drive.
- Analog comparator module with:
  - Two analog comparators.
  - Programmable on-chip voltage reference (V_{REF}) module.
  - Selectable internal or external reference.
  - Comparator outputs are externally accessible.
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler.
- Timer1: 16-bit timer/counter with external crystal/clock capability.
- Timer2: 8-bit timer/counter with 8-bit period registers prescaler and postscaler.
- Capture, Compare, PWM module:
  - 16-bit Capture/Compare.
  - 10-bit PWM.
- Addressable Universal Synchronous/Asynchronous Receiver/Transmitter USART/SCI.

Table 1-1: PIC 16F627A/628A/648A FAMILY OF DEVICES

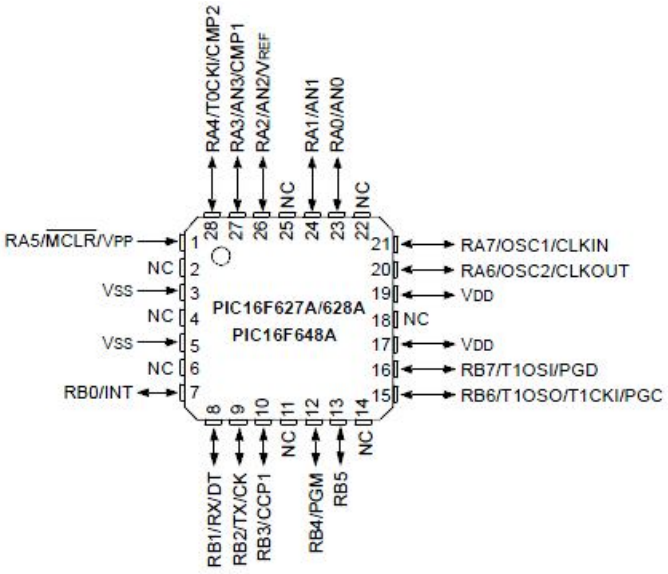
Device	Program Memory	Data Memory		I/O	CCP	USART	Comparators	timers
	Flash (words)	SRAM (bytes)	EEPROM (bytes)		PWM			8/16-bit
PIC 16F627A	1024	224	128	16	1	Y	2	2/1
PIC 16F628A	2048	224	128	16	1	Y	2	2/1
PIC 16F648A	4096	256	256	16	1	Y	2	2/1

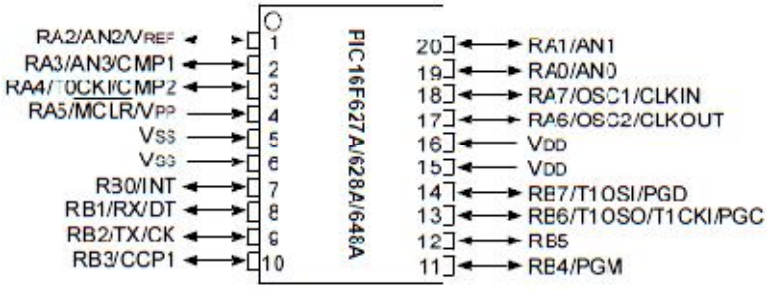
Pin Diagrams

PDIP, SOIC



28-Pin QFN


RA4/T0CKI/CMP2
RA3/AN3/CMP1
RA2/AN2/VREF
RA1/AN1
RA0/AN0
NC
NC
NC
RA5/MCLR/VPP
NC
VSS
NC
VSS
NC
RB0/INT
PIC16F627A/628A
PIC16F648A
RA7/OSC1/CLKIN
RA6/OSC2/CLKOUT
VDD
NC
VDD
RB7/T1OSI/PGD
RB6/T1OSO/T1CKI/PGC
RB1/RX/DT
RB2/TX/CK
RB3/CCP1
NC
RB4/PGM
RB5
NC


RA2/AN2/VREF
RA3/AN3/CMP1
RA4/T0CKI/CMP2
RA5/MCLR/VPP
VSS
VSS
RB0/INT
RB1/RX/DT
RB2/TX/CK
RB3/CCP1
PIC16F627A/628A/648A
RA1/AN1
RA0/AN0
RA7/OSC1/CLKIN
RA6/OSC2/CLKOUT
VDD
VDD
RB7/T1OSI/PGD
RB6/T1OSO/T1CKI/PGC
RB5
RB4/PGM

GENERAL DESCRIPTION

The PIC16F627A/628A/648A are 18-pin Flash-based members of the versatile PIC16F627A/628A/648A family of low –cost, high performance, CMOS, fully-static, 8-bit microcontrollers.

All PIC microcontrollers employ an advanced RISC architecture. The PIC16F627A/628A/648A has enhanced core features, an eight-level deep stack, and multiple internal interrupt sources. The separate instruction and data buses of the Harvard architecture allow a 14-bit wide instruction word with the separate 8-bit wide data.

The two-stage instruction pipeline allows all instructions (reduced instruction set) are available, complemented by a large register set.

PIC16F627A/628A/648A microcontrollers typically achieve a 2:1 code compression and 4:1 speed improvement over other 8-bit microcontrollers in their class.

PIC16F627A/628A/648A devices have integrated features to reduce external components, thus reducing system cost, enhancing system reliability and reducing power consumption.

The PIC16F627A/628A/648A has 8 oscillator configurations. The single-pin RC oscillator minimizes power consumption, XT is a standard crystal, and INTOSC is a self-contained precision two-speed internal oscillator.

The HS mode is for High-Speed crystal. The EC mode is for an external clock source.

The Sleep (Power-down) mode offers power savings. Users can wake-up the chip from Sleep through several external interrupts, internal interrupts and resets.
A highly reliable Watchdog Timer with its own on-chip RC oscillator provides protection against software lock-up.

Table 1-1 shows the features of the PIC16F627A/628A/648A mid-range microcontroller family.

A simplified block diagram of the PIC16F627A/628A/648A is show in figure 3-1.
The PIC16F627A/628A/648A series fits in applications ranging from battery chargers to low power remote sensors. The Flash technology makes customizing application programs (detection levels, pulse generation, timers, etc) extremely fast and convenient. The small footprint packages make this microcontroller series ideal

for all applications with space limitations. Low cost, low power, high performance, ease of use and I/O flexibility make the PIC16F627A/628A/648A very versatile.

Development Support

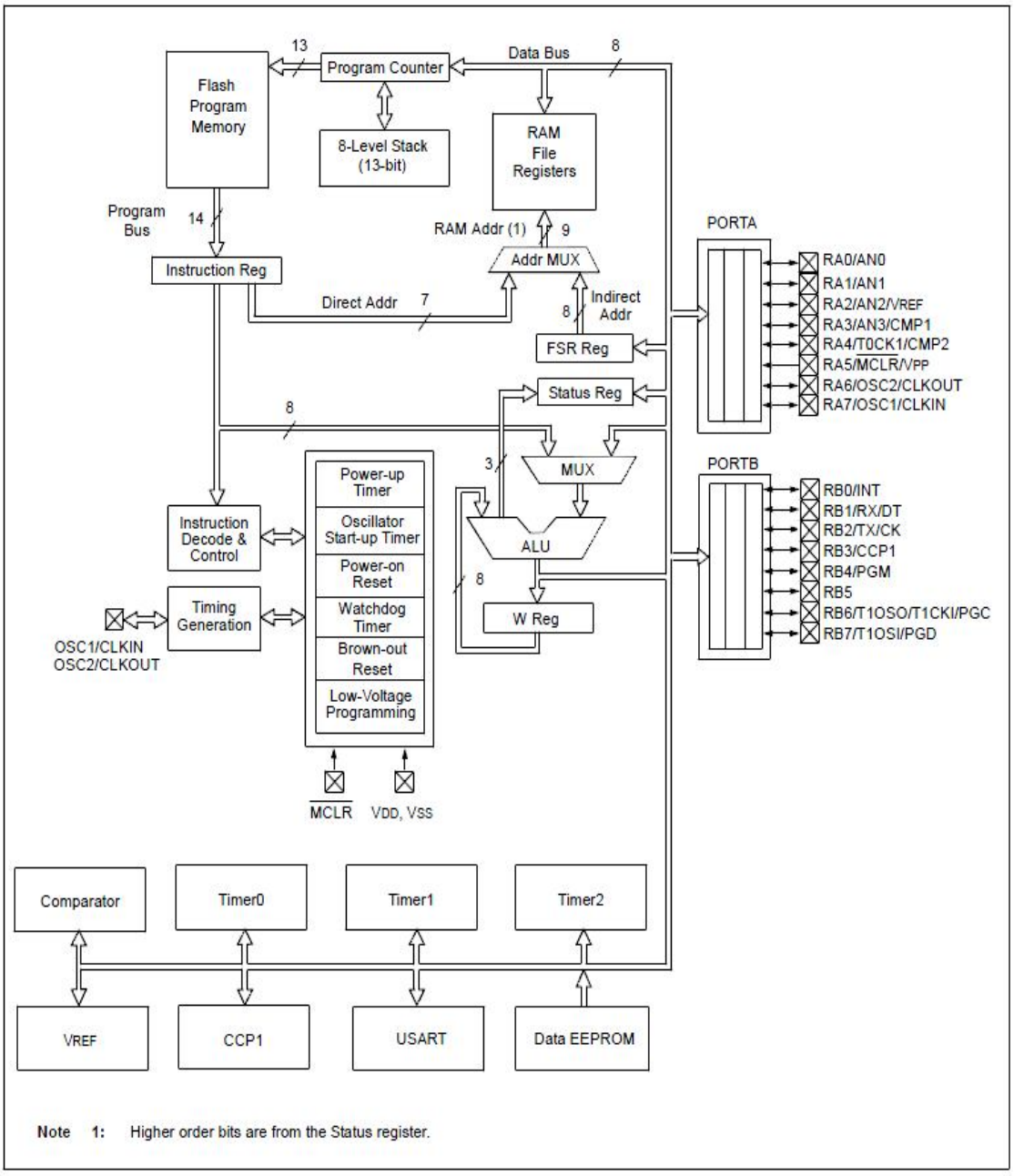
The PIC16F627A/628A/648A family is supported by a full-featured macro assembler, a software simulator, an in-circuit debugger, a low cost development programmer and a full-featured programmer. A third Party “C” compiler support tool is also available

Tabla 1-1

		PIC16F627A	PIC16F628A	PIC16F648A	PIC16LF627A	PIC16LF628A	PIC16LF648A
CLOCK	Maximum Frequency of Operation (MHz)	20	20	20	20	20	20
Memory	Flash Program Memory(words)	1024	2048	4096	1024	2048	4096
	RAM Data Memory (bytes)	224	224	256	224	224	256
	EEPROM Data Memory(bytes)	128	128	256	128	128	256
Peripherals	Timer Module(s)	TMR0,TMR1, TMR2	TMR0,TMR1, TMR2	TMR0,TMR1, TMR2	TMR0,TMR1, TMR2	TMR0,TMR1, TMR2	TMR0,TMR1, TMR2
	Comparator	2	2	2	2	2	2
	Capture/Compare/ PWM modules	1	1	1	1	1	1
	Serial Communicatios	USART	USART	USART	USART	USART	USART
	Internal Voltage Reference	YES	YES	YES	YES	YES	YES
Features	Interrupt Sources	10	10	10	10	10	10
	I/O Pins	16	16	16	16	16	16
	Voltage Range (Volts)	3,0-5,5	3,0-5,5	3,0-5,5	2,0-5,5	2,0-5,5	2,0-5,5
	Brown-out Reset	YES	YES	YES	YES	YES	YES
	Packages	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN

All PIC family devices have Power-on Reset, selectable Watchdog Timer, selectable code-protect and high I/O current capability.
All PIC 16F627A/628A/648A family devices use serial programming with clock pin RB6 and data pin RB7

FIGURE 3-1 BLOCK DIAGRAM



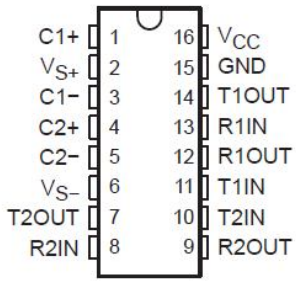
Note 1: Higher order bits are from the Status register.

ANEXO C: Especificaciones del MAX232

DUAL EIA-232 DRIVERS/RECEIVERS

- Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28
- Operates From a Single 5-V Power Supply With 1.0-μF Charge-Pump Capacitors
- Operates Up To 120 kbit/s
- Two Drivers and Two Receivers
- ±30-V Input Levels
- Low Supply Current . . . 8 mA Typical
- ESD Protection Exceeds JESD 22
  - 2000-V Human-Body Model (A114-A)
- Upgrade With Improved ESD (15-kV HBM) and 0.1-μF Charge-Pump Capacitors is Available With the MAX202
- Applications
  - TIA/EIA-232-F, Battery-Powered Systems, Terminals, Modems, and Computers

MAX232 . . . D, DW, N, OR NS PACKAGE
MAX232I . . . D, DW, OR N PACKAGE
(TOP VIEW)



Description/ordering information

The MAX 232 is dual driver/receiver that includes a capacitive voltage generator to supply TIA/EIA-22-F voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels.

These receivers have a typical threshold of 1.3V, a typical hysteresis of 0.5V, and accept ±30-V inputs.

Each driver converts TTL/CMOS input levels into TIA/EIA-232-F levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP (N)	Tube of 25	MAX232N	MAX232N
	SOIC (D)	Tube of 40	MAX232D	MAX232
		Reel of 2500	MAX232DR	
	SOIC (DW)	Tube of 40	MAX232DW	MAX232
		Reel of 2000	MAX232DWR	
	SOP (NS)	Reel of 2000	MAX232NSR	MAX232
−40°C to 85°C	PDIP (N)	Tube of 25	MAX232IN	MAX232IN
	SOIC (D)	Tube of 40	MAX232ID	MAX232I
		Reel of 2500	MAX232IDR	
	SOIC (DW)	Tube of 40	MAX232IDW	MAX232I
		Reel of 2000	MAX232IDWR	

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

Function Tables

EACH DRIVER

INPUT TIN	OUTPUT TOUT
L	H
H	L

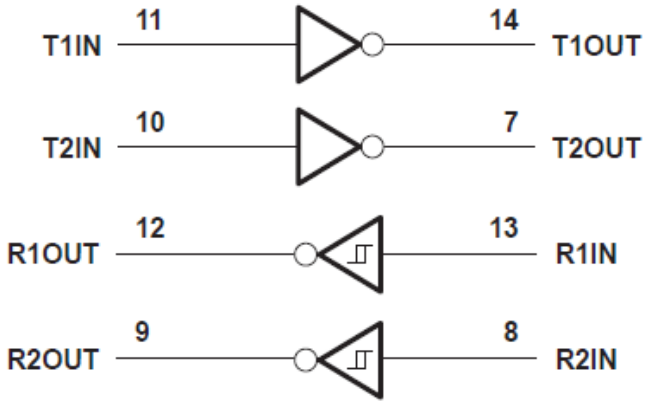
H = high level, L = low level

EACH RECEIVER

INPUT RIN	OUTPUT ROUT
L	H
H	L

H = high level, L = low level

Logic diagram (positive logic)



Absolute máximum ratings over operating free-air temperatura range (unless otherwisw noted)

Input supply voltage range, V_{CC} (see Note 1)	−0.3 V to 6 V
Positive output supply voltage range, V_{S+}	V_{CC} − 0.3 V to 15 V
Negative output supply voltage range, V_{S-}	−0.3 V to −15 V
Input voltage range, V_I: Driver	−0.3 V to V_{CC} + 0.3 V
Receiver	±30 V
Output voltage range, V_O: T1OUT, T2OUT	V_{S-} − 0.3 V to V_{S+} + 0.3 V
R1OUT, R2OUT	−0.3 V to V_{CC} + 0.3 V
Short-circuit duration: T1OUT, T2OUT	Unlimited
Package thermal impedance, θ_{JA} (see Notes 2 and 3): D package	73°C/W
DW package	57°C/W
N package	67°C/W
NS package	64°C/W
Operating virtual junction temperature, T_J	150°C
Storage temperature range, T_{stg}	−65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES:
1. All voltages are with respect to network GND.
2. Maximum power dissipation is a function of $T_J(max)$, θ_{JA}, and T_A. The maximum allowable power dissipation at any allowable ambient temperature is $P_D = (T_J(max) - T_A)/\theta_{JA}$. Operating at the absolute maximum T_J of 150°C can affect reliability.
3. The package thermal impedance is calculated in accordance with JESD 51-7.

Recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	V
V_{IH}	High-level input voltage (T1IN,T2IN)		2			V
V_{IL}	Low-level input voltage (T1IN, T2IN)				0.8	V
R1IN, R2IN	Receiver input voltage				±30	V
T_A	Operating free-air temperature	MAX232	0		70	°C
		MAX232I	−40		85	

Electrical characteristics over recommended ranges of suplly voltaje and operating free-air temperature (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP‡	MAX	UNIT
I_{CC}	Supply current	V_{CC} = 5.5 V, All outputs open, T_A = 25°C		8	10	mA

‡ All typical values are at V_{CC} = 5 V and T_A = 25°C.

NOTE 4: Test conditions are C1−C4 = 1 μF at V_{CC} = 5 V ± 0.5 V.

```
#include <16f628a.h>
#include <string.h>

#fuses intrc_io, wdt, noput, nomclr, nobrownout, nolvp, nocpd, noprotect
#use delay(clock=4M)

#priority INT_TIMER1,int_EXT

//prototipos de funciones
unsigned int16 CRC4_obtener(unsigned int8 dato);
unsigned int8 CRC4_get_dato(unsigned int16 dato_CRC);
unsigned int8 CRC4_revisar(unsigned int16 dato_CRC);

void secuencia_apagado(void);

void luces_frente(void);
void luces_traseras(void);
void direccional_izquierdo(void);
void direccional_derecho(void);

unsigned int8 get_string_overtime(char* s, unsigned int8 max);

//declaracion de constantes

#define generadorCRC4 0x13 //polinomio del CRC4 ITU-T G704 = x^4 + x + 1

#define HMTR_DTX     PIN_B1
#define HMTR_DRX     PIN_B2

#define COD_LUCES_ADE    0x80
#define COD_LUCES_ATR    0x40
#define COD_DIREC_IZQ    0x20
#define COD_DIREC_DER    0x10
#define COD_MOVER_ADE    0x08
#define COD_MOVER_ATR    0x04
#define COD_GIRAR_IZQ    0x02
#define COD_GIRAR_DER    0x01

#define LED_ON       PIN_A0
#define LED_FRENTE   PIN_A7
#define LED_ATRAS    PIN_A6
#define LED_IZQUI    PIN_B5
#define LED_DEREC    PIN_B6

#define MOT_ADE      PIN_A1 //color amarillo
#define MOT_ATR      PIN_A2 //color verde
#define MOT_DER      PIN_A3//color negro
#define MOT_IZQ      PIN_A4//color azul
```

```
#define PB_ON        PIN_B0

#define LEDON_ON output_high(LED_ON)
#define LEDON_OFF output_low(LED_ON)
#define LEDFRENTE_ON output_high(LED_FRENTE)
#define LEDFRENTE_OFF output_low(LED_FRENTE)
#define LEDATRAS_ON output_high(LED_ATRAS)
#define LEDATRAS_OFF output_low(LED_ATRAS)
#define LEDIZQ_ON output_high(LED_IZQUI)
#define LEDIZQ_OFF output_low(LED_IZQUI)
#define LEDDER_ON output_high(LED_DEREC)
#define LEDDER_OFF output_low(LED_DEREC)

#define MOTOR_ADE_ON {output_low(MOT_ATR); output_high(MOT_ADE); }
#define MOTOR_ATR_ON {output_low(MOT_ADE); output_high(MOT_ATR);}
#define MOTOR_DIR_OFF {output_low(MOT_ADE); output_low(MOT_ATR);}

#define MOTOR_IZQ_ON {output_low(MOT_DER);output_high(MOT_IZQ); }
#define MOTOR_DER_ON {output_low(MOT_IZQ); output_high(MOT_DER);}
#define MOTOR_GIR_OFF {output_low(MOT_IZQ); output_low(MOT_DER);}

#define CHAR_FINAL 0X08

#use RS232(BAUD=9600,XMIT=HMTR_DRX ,RCV=HMTR_DTX
,BITS=8,PARITY=N,ERRORS,TIMEOUT=5)//,DISABLE_INTS

//declaracion de variables globales
int iniciar=0;
unsigned int1 flag_led_on=0,flag_led_der=0,flag_led_izq=0,flag_led_fre=0,flag_led_atr=0;

int32 j;
int8 dato;

void main()
{
  char dato_CRC[20];
  unsigned int16 dato_CRC4;

  delay_ms(150);
  set_tris_a(0);
  output_a(0);

  set_tris_b(0x93);
  setup_comparator( NC_NC_NC_NC );
  LEDIZQ_OFF;
  LEDDER_OFF;

  ENABLE_INTERRUPTS( INT_EXT );
  ext_int_edge(H_TO_L);
  ENABLE_INTERRUPTS( GLOBAL );
  HMTREN_OFF;
```

```
MOTOR_DIR_OFF;
MOTOR_GIR_OFF;

j=0;
while(1)
{
  restart_wdt();
inicio:
  if(iniciar==1)
  {
    if(kbhit())
    {
      dato_CRC[0]=getc();
      dato_CRC[1]=getc();
      dato_CRC[2]=getc();
      getc();
    }
    else
    {
      goto inicio;
    }

    if(dato_CRC[2]==CHAR_FINAL)
    {
      dato_CRC4= (((unsigned int16)dato_CRC[0])<<8) | ((unsigned int16)dato_CRC[1]);
    }
    else
    {
      goto inicio;
    }


    if( CRC4_revisar(dato_CRC4)==1 )
    {
      dato = CRC4_get_dato(dato_CRC4);
      if(dato == COD_ESP)
        secuencia_apagado();
      else
      {
        if( (dato & COD_LUCES_ADE)!=0)
          luces_frente();

        if( (dato & COD_LUCES_ATR)!=0)
          luces_traseras();

        if( (dato & COD_DIREC_IZQ)!=0)
          direccional_izquierdo();

        if( (dato & COD_DIREC_DER)!=0)
          direccional_derecho();
```

```
            if( (dato & COD_MOVER_ADE)!=0)
            {   MOTOR_ADE_ON;}
            else if( (dato & COD_MOVER_ATR)!=0)
            {
              MOTOR_ATR_ON;
            }
            else
            {
              MOTOR_DIR_OFF;
            }

            if( (dato & COD_GIRAR_DER)!=0)
            {   MOTOR_DER_ON;}
            else if( (dato & COD_GIRAR_IZQ)!=0)
            {   MOTOR_IZQ_ON;}
            else
            {   MOTOR_GIR_OFF;}

            delay_ms(20);

            if(input(uSW_GOLPE))
            {
              printf("%c\r",0x01);
            }
            else
            {
             printf("%c\r",0);
            }
          }
        }
      }
    }
}

#int_EXT
void PB_detect()
{
  char code_pc[15],code_pic[15];
  code_pc[0]=0;
  clear_interrupt(INT_EXT);
  delay_ms(5);
  disable_interrupts(INT_EXT);
  delay_ms(5);
  disable_interrupts(GLOBAL);
  delay_ms(80);//100 ideal
  while(input(PB_ON));
  strcpy(code_pic,"STATUS_OK");

  if(iniciar==0)
  {
```

```
      iniciar=1;
      setup_uart(1);
      setup_uart(9600);
      HMTREN_ON;
      delay_ms(50);
      setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);
      enable_interrupts(INT_TIMER1);
      enable_interrupts(GLOBAL);
      flag_led_on=1;

      do{
        code_pc[0]=0;

        while( get_string_overtime(code_pc,14) != 1 )
          restart_wdt();

        if( strcmp(code_pc,code_pic) == 0 )
        {
          delay_ms(200);//110 funciona
          printf("CAR_ON\r");
          delay_ms(100);//110 funcion
        }
      }while( strcmp(code_pc,code_pic) != 0 );

      setup_timer_1(T1_DISABLED);
      disable_interrupts(INT_TIMER1);
      flag_led_on=0;
      flag_led_izq=0;

      LEDON_ON;
      clear_interrupt(INT_EXT);
      delay_ms(5);
      enable_interrupts(INT_EXT);
      delay_ms(5);
      enable_interrupts(GLOBAL);
      while(kbhit())//LIMPIA EL BUFFER DEL PIC
      getc();
      return;
    }
    else if(iniciar==1)
    {
  apagar:
      iniciar=0;
      printf("CAR_OFF\r");
      delay_ms(150);
      secuencia_apagado();
      setup_uart(0);
      delay_ms(50);
    }
    clear_interrupt(INT_EXT);
    delay_ms(5);
```

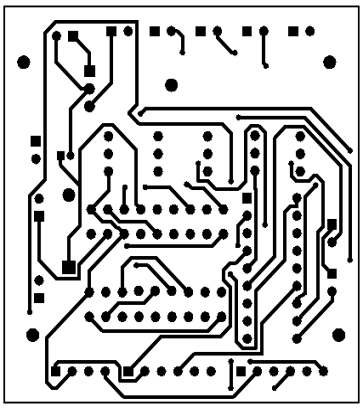
```
   enable_interrupts(INT_EXT);
   delay_ms(5);
   enable_interrupts(GLOBAL);
   return;
}

#INT_TIMER1
void temp(void)
{
   if(flag_led_on==1)
      output_toggle(LED_ON);
   if(flag_led_der==1)
      output_toggle(LED_DEREC);
   if(flag_led_izq==1)
      output_toggle(LED_IZQUI);
   if(flag_led_fre==1)
      output_toggle(LED_FRENTE);
   if(flag_led_atr==1)
      output_toggle(LED_ATRAS);
}

void secuencia_apagado(void)
{
   iniciar=0;
   delay_ms(10);
   setup_timer_1(T1_DISABLED);
   disable_interrupts(INT_TIMER1);

   LEDON_OFF;
   HMTREN_OFF;
   LEDIZQ_OFF;
   LEDDER_OFF;
   LEDFRENTE_OFF;
   LEDATRAS_OFF;
   MOTOR_DIR_OFF;
   MOTOR_GIR_OFF;
 }

void luces_frente(void)
{
   output_low(LED_ATRAS);
   output_toggle(LED_FRENTE);
}

void luces_traseras(void)
{
   output_low(LED_FRENTE);//
   output_toggle(LED_ATRAS);
}

void direccional_izquierdo(void)
```

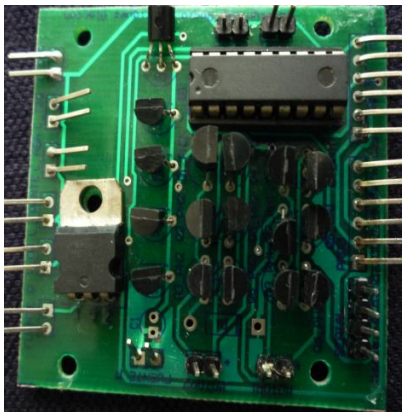
```
{
  flag_led_izq=~flag_led_izq;
  LEDDER_OFF;
  flag_led_der=0;
  if(flag_led_izq!=0)
  {
    setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);
    enable_interrupts(INT_TIMER1);
 }
  else
  {
    setup_timer_1(T1_DISABLED);
    disable_interrupts(INT_TIMER1);
    LEDIZQ_OFF;
  }
}

void direccional_derecho(void)
{
  flag_led_der=~flag_led_der;
  LEDIZQ_OFF;
  flag_led_izq=0;
  if(flag_led_der!=0)
  {
    setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);
    enable_interrupts(INT_TIMER1);
  }
  else
  {
    setup_timer_1(T1_DISABLED);
    disable_interrupts(INT_TIMER1);
    LEDDER_OFF;
  }
}

unsigned int8 CRC4_get_dato(unsigned int16 dato_CRC)
{
  return(dato_CRC>>8);
}

unsigned int8 CRC4_revisar(unsigned int16 dato_CRC)
{
  unsigned int8 CRC,CRC_calc;
  unsigned int16 dato;

  dato = dato_CRC>>8;
  CRC = dato_CRC;
  CRC_calc=((dato<<4) % generadorCRC4);
  if( CRC ==  CRC_calc )
  {
    return(1);
```

```
  }
  return(0);
}

unsigned int8 get_string_overtime(char* s, unsigned int8 max)
{
  unsigned int16 len=0;
  unsigned int8 flag_leido;
  unsigned int16 timer=0;
  char c;

  len=0;
  flag_leido=0;
  s[len]=0;
  if(!kbhit())
    return(0);

  do
  {
    c=255;
    flag_leido=0;

    for(timer=0;timer<=OVERTIME && c==255;timer++)
    {
      if(kbhit())
      {
        c=getc();
        flag_leido=1;
      }
    }
    if(flag_leido==0)
    {
      s[0]=0;
      return(2);
    }
    if ((c>=' ')&&(c<='~'))
    {
      if(len<=max)
      {
        s[len++]=c;
      }
      else
      {
        s[max+1]=0;
        return(3);
      }
    }
  }while(c!= 0X0D);
  s[len++]=0;
  return(1);
}
```

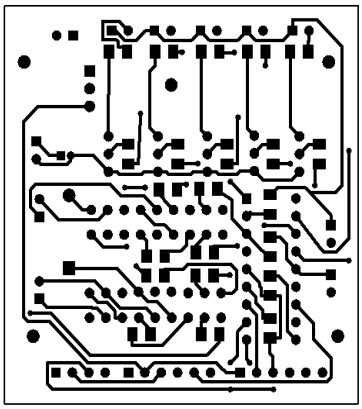
ANEXO E

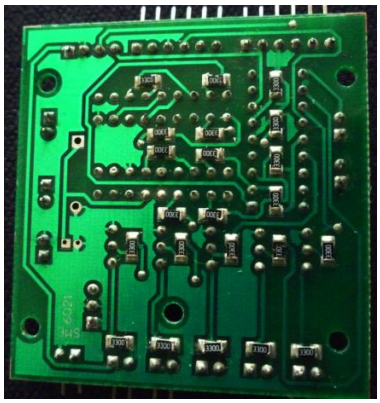
CIRCUITO IMPRESO
DEL MECANISMO TELEOPERADO



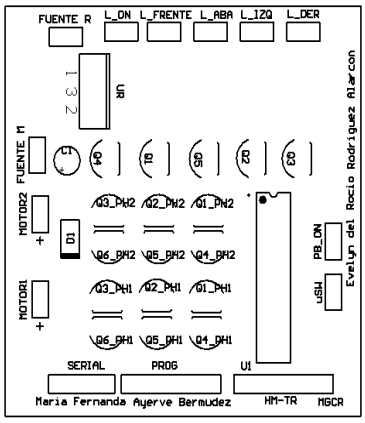


CAPA SUPERIOR





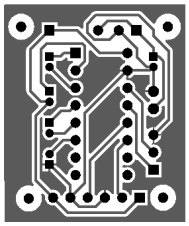
CAPA INFERIOR



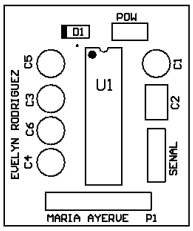
IMPRESIÓN SOBRECARA

ANEXO F

CIRCUITO IMPRESO
CONVERTIDOR SERIAL RS232 A SERIAL TTL



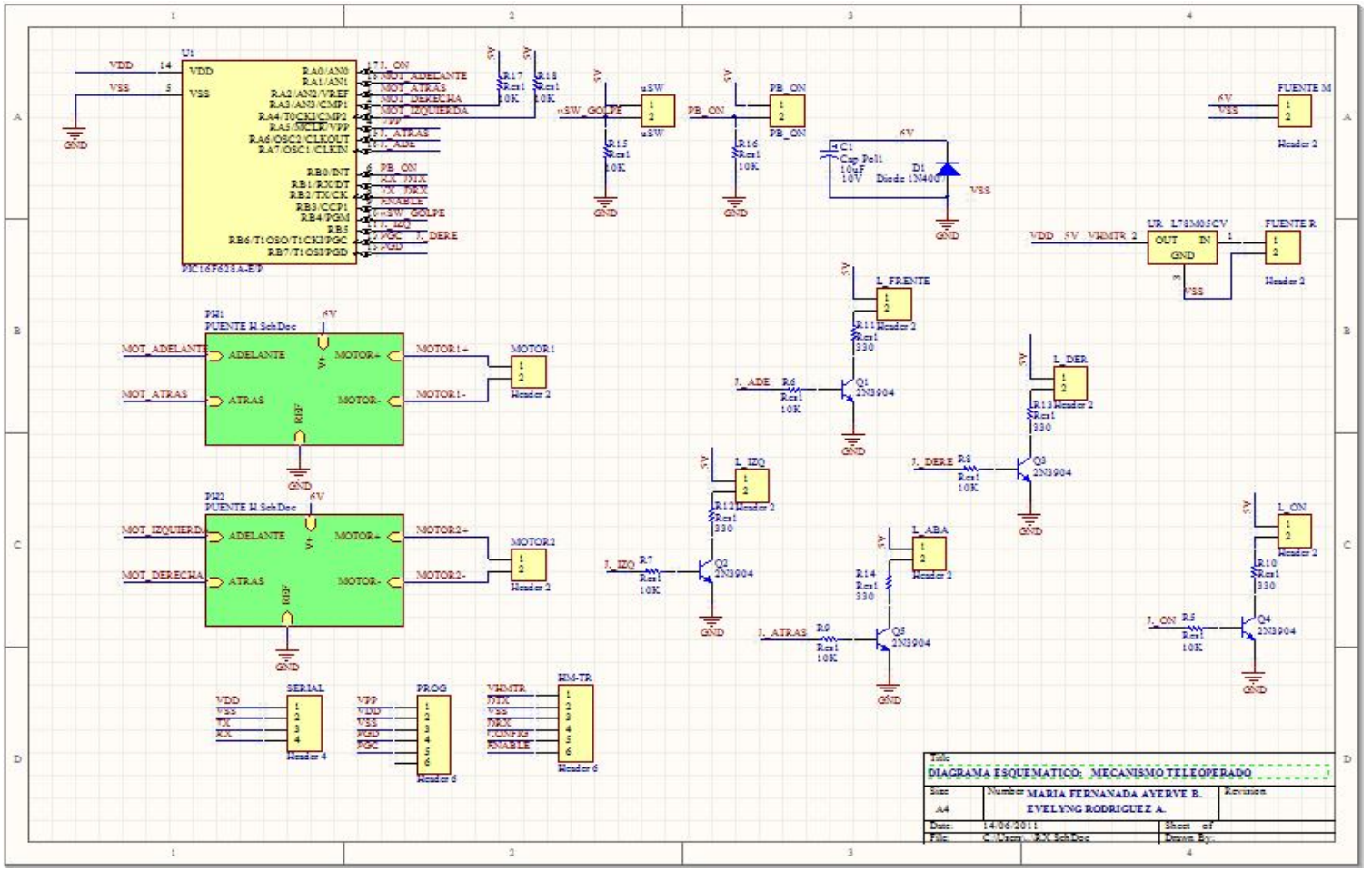
CARA INFERIOR



IMPRESIÓN SOBRECARA

ANEXO G

DIAGRAMA ESQUEMÁTICO MECANISMO TELEOPERADO



BIBLOGRAFíA

1. Bishop Robert H., LabVIEW 2009 Student Edition, Pearson, 2010
2. Console plus, Usar El Wiimote en el PC version1.1, http://consoleplus.wordpress.com/2007/07/08/usar-el-wiimote-en-el-pc-version11, Agosto 2010.
3. Wikipedia, Wiimote, http://es.wikipedia.org/wiki/Wiimote, Agosto 2010.
4. Wiibrew, Wiimote, http://wiibrew.org/wiki/Wiimote, Agosto 2010.
5. Bluetooth, Technical Information, http://bluetooth.com/Pages/Tech-Info.aspx, Agosto 2010.
6. Wikipedia, Bluetooth, http://es.wikipedia.org/wiki/Bluetooth , agosto 2010.
7. Eveliux, Bluetooth afila sus dientes, http://eveliux.com/mx/Bluetooth-afila-sus-dientes.php, Agosto 2010.
8. Sandra LabVIEW, definición, http://sandraLabVIEW.blogspot.com/2008/06/definicin.html, Agosto 2010
9. 3D Juegos, Wiimote, http://www.3djuegos.com/foros/index.php?id_tema=2416599&page=0&marca_mensaje=2416599&saltos=1, Agosto 2010
10. Wikipedia, Cyclic redundancy check, http://en.wikipedia.org/wiki/Cyclic_redundancy_check, Agosto 2010