UNIVERSIDAD POLITECNICA SALESIANA

SEDE QUITO – CAMPUS SUR

CARRERA DE INGENIERIA ELECTRONICA

MENCION TELECOMUNICACIONES

DISEÑO E IMPLEMENTACION DE UN MODULO DE LABORATORIO PARA LOS CODIGOS DE LINEA, EMPLEANDO MICROCONTROLADORES CON UNA INTERFAZ USB DE CONEXIÓN A LABIEW PARA DISEÑO, CONTROL Y PRUEBAS DE CODIFICACION Y DECODIFICACION DE DATOS.

TESIS PREVIA A LA OBTENCIÓN DEL TITULO DE INGENIERO EN ELECTRÓNICA MENCION TELECOMUNICACIONES

CRISTIAN FERNANDO ZAPATA YUGSI
cris_zapata131011@hotmail.com

FRANCISCO XAVIER JURADO PRUNA
franciscojuradop@gmail.com

DIRECTOR: Ing. José Antonio Pazmiño

jpazmino@ups.edu.ec

Quito, Marzo 2012

DECLARACIÓN

Nosotros, Cristian Fernando Zapata Yugsi y Francisco Xavier Jurado Pruna, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría, que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Politécnica Salesiana, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Cristian Fernando Zapata Yugsi
171886130-0

Francisco Xavier Jurado Pruna
171549437-1

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Cristian Fernando Zapata Yugsi y Francisco Xavier Jurado Pruna, bajo mi supervisión.

Ing. José Antonio Pazmiño

DIRECTOR DE PROYECTO

AGRADECIMIENTO

Primero y antes que nada, doy gracias a Dios, por darme la vida y haberme dado salud para lograr mis objetivos propuestos, además de siempre darme su infinito amor.

A mi madre Marlene Yugsi por su amor y apoyo incondicional y por brindarme su ayuda cuando más la necesitaba y enseñarme que todo esfuerzo a la final es recompensado.

A mi esposa ser mi compañera inseparable en los momentos malos como buenos y seguir a mi lado, y a todos aquellos que participaron directa o indirectamente en la elaboración de esta tesis.

A mi hermana Andrea por su cariño y seguir el ejemplo brindado.

A toda mi familia por ser el respaldo incondicional en mi, qué gracias a sus consejos y palabras de aliento crecí como persona, un agradecimiento especial a mi abuelita Elvia que fue mi apoyo incondicional y haberme brindado todos sus concejos sabios .

A mis grandes amigos por estar siempre conmigo apoyándome en todo las circunstancias posibles, también son parte de esta alegría.

A todas aquellas personas que de una u otra forma, colaboraron o participaron en la realización de este proyecto, mis más sinceros agradecimientos.

¡Gracias a ustedes!

Cristian Fernando Zapata Yugsi

DEDICATORIA

Primero doy gracias a Dios por permitirme llegar a este momento tan especial en mi vida, por estar en los momentos malos como en los momentos buenos como este.

A mi madre, a mi familia por haberme educado y enseñarse su tenacidad, lucha y perseverancia , que a sido el pilar fundamental en mi vida ya que gracias a ellos soy lo que soy hoy en día; a mi hermana por el cariño que me brinda y a mi esposa que me a dado los ánimos para seguir luchando por un futuro mejor.

Cristian Fernando Zapata Yugsi

AGRADECIMIENTO

En primer lugar a Dios por haberme guiado en todo momento durante todo el largo trayecto que he recorrido hasta poder culminar mi carrera, en segundo lugar a mi madre por siempre haberme dado su fuerza y apoyo incondicional que me han ayudado y llevado hasta donde estoy ahora. Por último a mi compañero de tesis porque hemos logrado culminar con éxito y con mucho esfuerzo nuestro proyecto de titulación y a mi director de tesis quién supo ayudarnos en todo momento, Ingeniero José Pazmiño.

Francisco Xavier Jurado Pruna

DEDICATORIA

Dedico este trabajo a mi madre quien siempre me apoyo en cada paso que he dado, me ha cuidado y me ha enseñado a buscar la solución a cada problema me ha impulsado en los momentos difíciles durante el transcurso de mi carrera y que gracias a ella hoy puedo llegar a culminar una etapa muy importante de mi vida.

Francisco Xavier Jurado Pruna

UNIVERSIDAD POLITECNICA SALESIANA
SEDE QUITO – CAMPUS SUR

CARRERA DE INGENIERIA ELECTRONICA
MENCION TELECOMUNICACIONES

ABSTRACT

CRISTIAN FERNANDO ZAPATA YUGSI
cris_zapata131011@hotmail.com
FRANCISCO XAVIER JURADO PRUNA
franciscojuradop@gmail.com

Quito, Marzo 2012

En Las Telecomunicaciones, los códigos de línea o también conocidos como modulación en banda base son códigos utilizados en un sistema de comunicación para propósitos de transmisión.

Los códigos de línea son frecuentemente usados para el transporte digital de datos, estos códigos consisten en representar una señal digital transportada respecto a su amplitud respecto al tiempo. Esta señal se encuentra perfectamente sincronizada debido a las propiedades específicas que posee la capa física. La representación de la onda se la suele realizar mediante un número determinados impulsos. Estos impulsos representan los 1s y los 0s digitales.

Después de la codificación en línea, la señal es enviada a través de la capa física. A veces las características de dos canales aparentemente muy diferentes son lo suficientemente parecidos para que el mismo código sea usado por ellos.

El presente proyecto de titulación consiste en el desarrollo de un equipo para ser utilizado en el laboratorio construido con materiales fáciles de encontrar en nuestro medio complementado con los conocimientos adquiridos durante el transcurso de la carrera, con el objetivo que este equipo sirva para reforzar los conocimientos en el tema de la codificación de línea que se imparte en la materia de Comunicaciones Digitales, mostrando de una forma práctica y con la ayuda de un software para la simulación creado en LabView el funcionamiento de los diferentes algoritmos de codificación aquí implementados.

El proyecto que se va a explicar se lo titulo como **DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DE LA LABORATORIO PARA LOS CÓDIGOS DE LÍNEA, EMPLEANDO MICROCONTROLADORES CON UNA INTERFAZ USB PARA CONEXIÓN A LABIEW PARA DISEÑO, CONTROL Y PRUEBAS DE CODIFICACIÓN Y DECODIFICACIÓN DE DATOS.**

El objetivo principal planteado fue “Diseñar e implementar un módulo didáctico para mostrar ocho clases de códigos de línea en las prácticas de laboratorio, mediante el empleo de Microcontroladores PIC 18F4550 y el software de LabView” con la finalidad de que este equipo sea utilizado en el laboratorio como complemento al aprendizaje de los códigos de línea en la materia de Comunicaciones Digitales

Se escogió el tema de códigos de línea debido a que tienen diversas e importantes aplicaciones en lo referente al transporte digital de datos en un medio guiado como por ejemplo los código NRZ o No Retorno a cero son utilizados para la grabación de cintas magnéticas, también tenemos los códigos AMI que fueron utilizados en la primera generación de redes PCM y actualmente aún se los encuentra en algunos equipos multiplexores, los códigos Diferenciales son utilizados en transmisiones asíncronas, también existen los códigos Manchester los cuales se los utiliza en las redes LAN con diferentes topologías.

El equipo desarrollado consta de dos partes el modulo codificador y el modulo decodificador los cuales poseen las siguientes características.

Modulo Codificador

- Permite al usuario que ingrese los datos a ser codificados a través de una computadora mediante conexión usb o por el teclado que tiene incorporado.
- Permite el envío de datos desde 1 a 8 bytes.

- El modulo tiene una salida de conexión para el osciloscopio con el objetivo de poder visualizar las codificación de los datos.

Básicamente este equipo consta de un PIC para realizar el proceso de codificación de los datos un teclado para el ingreso de los mismos, una pantalla LCD en la cual se visualizaran los diferentes parámetros que se van ingresando para el proceso de codificación y la respectiva conexión USB para poder interactuar con el software respectivo.

EL diagrama de bloques reducido por motivos de la presentación correspondiente al microcontralador del módulo codificador posee el inicio del programa, el proceso de codificación en donde se realizan las declaraciones de variables y las diferentes funciones que realizan la codificación de linea, y en el programa principal se realizan las llamadas a estas funciones dependiendo de cómo se realice el ingreso de los datos que puede vía usb por la interfaz gráfica o por el teclado.

El módulo codificador del presente trabajo se lo ensamblo en base al diagrama esquemático mostrado, donde se puede observar el circuito correspondiente a la fuente de energía que sirve de alimentación del circuito y proporciona los niveles de voltaje positivos y negativos necesarios para realizar la codificación de línea, además de las conexiones de los puertos del PIC18F4550 para el manejo del LCD, teclado y conexión USB.

Modulo Decodificador

- Permite al usuario seleccionar el código de línea de acuerdo al utilizado en el codificador.
- El modulo tiene conexión una conexión mediante el puerto usb para visualizar los datos decodificados en el software de Lab View.

La forma en que los datos que se presentan al usuario a través del LCD es en formato hexadecimal.
Este equipo consta de un PIC para realizar el proceso de decodificación de los datos recibidos desde el modulo codificador,

EL diagrama de bloques reducido por motivos de la presentación correspondiente al microcontralador del módulo decodificador posee el inicio del programa, el proceso de decodificación en donde se realizan las declaraciones de variables, las diferentes funciones que cumplen con los algoritmos de decodificación de linea y en el programa principal se realizan las llamadas a estas funciones para poder visualizar los datos recibidos en el LCD.

El Software del presente proyecto el cual permitirá desde la computadora el manejo del módulo codificador y decodificador de códigos de línea se lo hizo con el software de LabView en su versión 8.5 debido a que este permite la creación de SubVIs los cuales so subprogramas que formaran parte del programa principal facilitando el desarrollo del mismo, esta característica se la utilizo principalmente para la creación del programa de codificador.

En el programa correspondiente al codificador se ingresan los datos correspondientes a la frecuencia , la selección del código a ser utilizado y la trama de datos a ser enviados los cuales pueden ser desde 1 byte hasta 8 bytes, en este programa se realiza la simulación de la codificación de los datos, mostrando los resultados en tres graficas:

La primera grafica que se muestra nos indica la forma de los bits que se enviaran, la segunda grafica corresponde a la señal codificada en función del tiempo y la tercera es la señal codificada en función de la frecuencia.

En el programa correspondiente al módulo decodificador se presentan los datos recibidos en el equipo, permitiendo realizar la selección del código a ser utilizado por el decodificador también desde el software.

Para comprobar el funcionamiento correcto del equipo se hizo una prueba con cada uno de los códigos de línea implementados verificando que las gráficas obtenidas mediante la visualización en el osciloscopio y la de la señal simulada sean similares.

INDICE

Índice de Figuras

Índice de Tablas

RESUMEN

En Las Telecomunicaciones, los códigos de línea o también conocidos como modulación en banda base son códigos utilizados en un sistema de comunicación para propósitos de transmisión.

Los códigos de línea son frecuentemente usados para el transporte digital de datos, estos códigos consisten en representar una señal digital transportada respecto a su amplitud respecto al tiempo. Esta señal se encuentra perfectamente sincronizada debido a las propiedades específicas que posee la capa física. La representación de la onda se la suele realizar mediante un número determinados impulsos. Estos impulsos representan los 1s y los 0s digitales. Los tipos más comunes de codificación en línea son el unipolar, polar, bipolar y Manchester.

Después de la codificación en línea, la señal es enviada a través de la capa física. A veces las características de dos canales aparentemente muy diferentes son lo suficientemente parecidos para que el mismo código sea usado por ellos.

CAPITULO I

Estudio del Problema

1.1 INTRODUCCION

En este capítulo, se dará una introducción sobre la motivación de la tesis, abordando una breve descripción actual de los diferentes equipos de laboratorio que se pueden encontrar en el mercado y que son similares al que se desea desarrollar en este proyecto adicionando las características de cada uno de ellos, junto con una descripción de los objetivos que se pretenden alcanzar con el desarrollo de este proyecto.

1.2 SITUACION ACTUAL

Actualmente los estudiantes que cursan el séptimo nivel de la Carrera de Ingeniería Electrónica en la Mención de Telecomunicaciones al momento de adquirir los diferentes conocimientos que existen en la materia de Comunicaciones Digitales en lo referente al tema de códigos de línea lo realizan únicamente de forma teórica, resultado su formación incompleta al no poder relacionar la parte teórica aprendida con su respectiva parte práctica.

Una solución es la adquisición de equipos de laboratorio los cuales permitan contribuir al aprendizaje de los Códigos de Línea ya que estos al ser desarrollados para mejorar las prestaciones de los sistemas de transmisión resulta ser un tema importante dentro de la formación académica de un Ingeniero en Electrónica en la mención de Telecomunicaciones. En la actualidad podemos encontrar varias empresas dedicadas a la fabricación de equipos para laboratorio los cuales son utilizados en la enseñanza de diversos temas dentro del campo de la Electrónica, entre algunos equipos similares al que va a ser desarrollado en este proyecto podemos citar los siguientes:

De la empresa Lucas-Nülle que se dedica desde hace treinta años al desarrollo, producción y distribución de sistemas de entrenamientos modernos utilizados en la formación profesional y el perfeccionamiento, encontramos al **"Curso de Telecomunicaciones 5: Métodos de modulación de impulsos PAM/PCM/Delta"** el cual se muestra en la **Figura 1,** con este módulo se podría realizar el estudio de la

codificación y decodificación de Línea aunque este equipo no es desarrollado exclusivamente para el estudio de este tema, el equipo presenta los siguientes contenidos:

- Conocimiento del funcionamiento de la modulación y demodulación PAM y del método múltiplex por división en el tiempo.
- Conocer el teorema de muestreo según Shannon
- Mediciones de la curva de las señales moduladas por el método PAM
- Filtración optima, antialiasing
- Conocimiento del funcionamiento de la modulación y demodulación PCM
- Enumeración de las ventajas y desventajas de la transmisión digital
- Conocimiento del principio de la cuantificación de señales analógicas
- Formación de códigos paralelos y serie
- Método de compresión / expansión de acuerdo con la ley A y la ley µ
- Determinación por medición técnica del intervalo de cuantificación en la transmisión PCM
- Medición de la curva de las señales codificadas por el método PCM
- Registro de las características logarítmicas de transmisión con compansión según la ley a y la ley µ (13 ó 15 segmentos)
- Medición de la curva de una señal PCM multiplexada por división de tiempo
- Conocimiento de las propiedades más importantes del código de línea
- Conocimiento de los códigos pseudoternarios: Códigos AMI, HDB3 y AMI modificado
- Medición de la curva de señales con código de línea: Códigos AMI, HDB3 y AMI modificado
- Transmisión de la información, a través de una línea, conjuntamente con los datos de reloj y sincronización
- Recuperación de sincronización y fluctuación de fase;
- Conocimiento del formato de trama de datos del nivel 1 en RDSI (red digital de servicios integrados)
- Nivel 1 de la RDSI, análisis del marco de datos y de los bits en lo referente a su posición y función
- Análisis de una transmisión en 2 canales con canal de control y una velocidad de transmisión de datos de 64kbits por canal

Figura 1.1: Equito de Laboratorio Lucas-Nülle
Fuente **http://es.lucas-nuelle.com**

De la empresa EDIBON, en Madrid España dedicada al suministro de Equipos Didácticos Técnicos en lo que se refiere al tema de codificación y decodificación de línea posee un equipo dentro del tema de comunicaciones Digitales denominado **"Laboratorio Integrado de Comunicaciones"** el cual se muestra en la **Figura 2,** presenta las siguientes características:

- Tarjeta de emisión (EDICOM 5.1): para codificación de datos y modulación de señales.
- Tarjeta de recepción (EDICOM 5.2): para la demodulación de señales y decodificación de datos.
- Estudio de las técnicas de modulación/demodulación por portadora: ASK, PSK, FSK y QPSK.
- Estudio de los formatos de codificación de datos: NRZ (L), NRZ (M), RZ, Bifase (Manchester), Bifase (Mark).
- Frecuencias onda portadora: 1,44 MHz, (I) 960 KHz, (Q) 960 KHz.
- Incluye dos moduladores de portadora y dos convertidores unipolar-bipolar.
- Elementos: un inversor de datos, amplificador-sumador, decodificadores de bit en el receptor.

Figura 1.2: Equipo de Laboratorio EDIBON

Fuente: **http://www.edibon.com**

Con las especificaciones anteriores del equipo se pueden realizar las siguientes prácticas:

1. Estudio de los códigos de línea utilizados en la transmisión digital a corta distancia: NRZ (L), NRZ (M), RZ, AMI, RB, Bifase Manchester, Bifase Mark.
2. Relación entre régimen binario y velocidad de modulación.
3. Análisis de las técnicas de modulación digital: ASK, PSK, FSK, QPSK, estudiando sus características en el transmisor y demodulación en el receptor.
4. Simulación de fallos.

Por último se citará a la empresa NUEVOS RECURSOS encargada de suministrar equipos dentro de las áreas de Ensayo de Materiales, Equipos Didácticos para la enseñanza, Equipos de Laboratorio y Alimentos, carrocerías especiales, dentro del tema de codificación y decodificación de línea encontramos al equipo denominado **"Transmisión de datos en Banda Base"** este panel permite el estudio de las técnicas de transmisión en banda base con diferentes tipos de codificación:

- RZ (return to Zero), NRZ (Not return to Zero), Manchester, Biphase y Duobinaria.

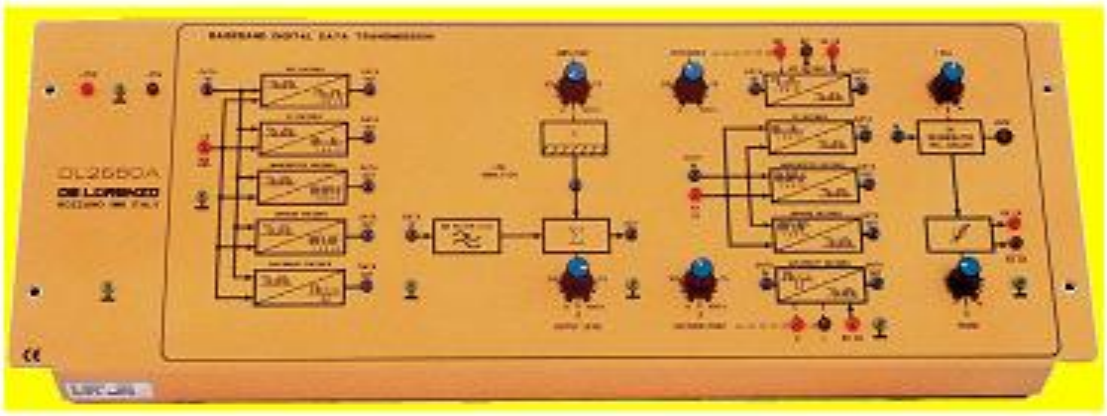


Figura 1.3: Equipo de Laboratorio de Nuevos Recursos
Fuente: **http://www.nuevosrecursos.com**

Lo descrito en los equipos anteriormente mencionados de las diferentes empresas fabricantes se puede comprobar al revisar sus características que son utilizados para mejorar la comprensión de los diferentes temas para los cuales fueron desarrollados entre ellos la codificación de línea, pero como se pudo verificar no existe un módulo

exclusivo para la codificación y decodificación de línea, adicionalmente el costo de uno de estos equipos es muy elevado.

Debido a que la codificación de línea es un método utilizado para adaptar o convertir la información representada en niveles lógicos TTL, CMOS, etc., encontrados en dispositivos digitales tales como computadoras, routers y equipos de comunicaciones a un formato de señal que sea adecuado para su transporte por un medio físico alámbrico, los Códigos de Línea son claves para la adecuada recepción de señales en ambientes puros digitales. Sus áreas de aplicación van desde las redes de datos locales hasta las redes de transporte, resultando adecuado tener un módulo de laboratorio para comprender de mejor forma este tema importante dentro del mundo de las comunicaciones.

1.3 OBJETIVO PRINCIPAL

Diseñar e implementar un módulo didáctico para mostrar ocho clases de códigos de línea en las prácticas de laboratorio, mediante el empleo de Microcontroladores PIC 18F4550 y el software de LabView.

1.4 OBJETIVOS ESPECIFICOS

- Esquematizar y desarrollar un módulo de laboratorio que muestre ocho técnicas de codificación y decodificación de línea.

- Elaborar un programa en el entorno gráfico del software de LabView para mostrar el funcionamiento del módulo.

- Realizar las diferentes pruebas de funcionamiento del equipo.

- Desarrollar una pequeña prueba al final de las prácticas para comprobar que los estudiantes comprendieron de mejor forma la codificación y decodificación de línea.

CAPITULO II

Viabilidad del proyecto

2.1 INTRODUCCION

El presente capítulo tiene por objeto, realizar un breve estudio de Prefactibilidad comercial, financiera y técnica para poder determinar la viabilidad de la elaboración del módulo de laboratorio planteado en el presente trabajo, para aportar con la comprensión de los estudiantes de la carrera de Ingeniería Electrónica en la Mención de Telecomunicaciones para el tema de codificación y decodificación de línea.

2.2 FACTIBILIDAD COMERCIAL

Este breve estudio nos sirve para poder determinar si el proyecto a ser desarrollado es de utilidad para los estudiantes en su proceso de formación, para lo cual en la factibilidad comercial se realizará el estudio de mercado mediante la aplicación de una encuesta a los estudiantes de octavo y décimo nivel de la Carrera.

2.2.1 ESTUDIO DE MERCADO

Este proyecto está desarrollado para el uso exclusivo de los alumnos que cursan séptimo semestre de la Carrera de Ingeniería Electrónica en la Mención de Telecomunicaciones de la Universidad Politécnica Salesiana debido a que en este nivel se recibe la materia de Comunicaciones Digitales y para la cual no se cuenta con un equipo de laboratorio para mostrar en forma práctica la codificación y decodificación de línea.

Para realizar este estudio se desarrollo una encuesta dirigida a todos los estudiantes de niveles superiores con el objetivo de examinar los conocimientos adquiridos por los estudiantes en el tema de códigos de línea para poder realizar el análisis de los resultados que influyeron en el desarrollo de este proyecto.

2.2.2 ANALISIS DE RESULTADOS

Para el desarrollo del presente análisis se utilizan los resultados obtenidos en base a las respuestas que proporcionaron a las diferentes preguntas planteadas en la

encuesta realizada a varios estudiantes, con la ayuda de la tabulación de los resultados y cuyas gráficas las podemos encontrar en el Anexo No 1, se obtuvieron los siguientes resultados:

En la pregunta número uno, referente a la clase de códigos de línea que se conoce, se dividió a los códigos de línea en cuatro grupos siendo estos el NRZ, AMI, Diferencial y Bifase para poder tabular los datos para su análisis respectivo de los cuales, un 34% de los encuestados respondió que conocía los códigos de línea NRZ, un 7% mencionó los códigos de línea AMI, un 2% mencionó los códigos de línea Diferencial y un 57% mencionó a los códigos de línea Bifase.

En la pregunta número dos, que se refiere al conocimiento de la definición de la codificación de línea, un 28% respondió que conocía que es la modulación de banda base y escribió una definición correcta mientras que el 72% respondió que no conocía lo que es la modulación de banda base o codificación de línea.

En la pregunta número tres, que se refiere a la identificación de un código de línea, un 48% respondió afirmativamente y contesto correctamente, mientras que el 52% no lo pudo identificar.

En la pregunta número cuatro, que se refiere a la utilización de los códigos de línea, un 80% respondió afirmativamente siendo esta la respuesta correcta a la pregunta, mientras que el 20% respondió que no.

En la pregunta número cinco, que se refiere a la utilización del código de línea Manchester, un 24% respondió afirmativamente siendo esta la respuesta correcta a la pregunta, mientras que el 76% respondió que no.

Y por último en la pregunta número seis planteada, referente a que si está de acuerdo con al desarrollo del módulo planteado en el presente trabajo, el 100% de los encuestados respondió afirmativamente.

Al realizar un análisis de los datos obtenidos en cada una de las preguntas de la encuesta podemos observar que los conocimientos adquiridos no se encuentran muy

claros, necesitan ser complementados, siendo una forma de hacerlo el relacionar la parte teórica aprendida con su respectiva parte práctica siendo de mucha utilidad el módulo propuesto en el presente trabajo.

2.3 FACTIBILIDAD FINANCIERA

Con la ayuda de este breve estudio definiremos si se dispone del financiamiento necesario para invertir en el desarrollo de este equipo, y si al compararlo con el costo de equipos similares alcanza a cumplir con las expectativas planteadas, para lo cual se realizó una comparación de precios de los equipos de laboratorio que podemos encontrar en el mercado de las diferentes empresas fabricantes y un aproximado del costo de fabricación que tendría el módulo propuesto.

Los equipos fabricados por las empresas Lucas- Nülle, EDIBON y Nuevos Recursos que únicamente distribuyen estos módulos a empresas o centros educativos tienen un costo aproximado que varía entre 900 a 1500 USD adicionalmente a este precio se debe agregar el costo del transporte, de impuestos por la importación estos valores varían dependiendo de la empresa debido al país en el que se encuentran y de diferentes elementos adicionales requeridos para su funcionamiento como lo indican los fabricantes de las diferentes empresas lo que supone rubros adicionales que aumentan su costo.

El equipo de laboratorio planteado en el presente trabajo alcanza un costo de fabricación de aproximadamente 700 USD. Al comparar estos datos se puede deducir que resulta mucho más económica la implementación del equipo planteado en el presente proyecto de titulación que el adquirir un módulo de laboratorio de cualquiera de las diferentes empresas antes mencionadas.

2.4 FACTIBILIDAD TECNICA

Este breve estudio de factibilidad Técnica nos sirve para determinar si se dispone de los conocimientos y habilidades necesarias en los diferentes procedimientos y funciones necesarias para el diseño e implementación del hardware y software del

módulo planteado en el presente proyecto, así como el costo de fabricación del mismo.

2.4.1 REQUERIMIENTOS DE HARDWARE

Para el desarrollo del hardware se utilizará el Microcontrolador 18F4550, este microcontrolador que es fabricado por MicroChip familia a la cual se le denomina PIC, posee varas características que lo hacen un dispositivo muy versátil y fácil de utilizar, al ser este dispositivo programable y capaz de realizar diferentes actividades relacionadas con el procesamiento de datos digitales, del control y comunicación digital de diferentes dispositivos, resulta una buena opción a ser utilizada en el presente proyecto.

Por lo que se requiere tener un buen conocimiento del manejo de los diferentes tipos de software existente y que se utilizará para poder programar el microcontrolador para que cumpla con las funciones requeridas para el buen funcionamiento del equipo a desarrollar.

2.4.2 REQUERIMIENTOS DE SOFTWARE

Para que un estudiante pueda interactuar con el hardware se requiere del desarrollo de una interfaz gráfica para lo cual se desarrollará un programa con la ayuda del software para diseño control y pruebas LabView el mismo que nos servirá para que los estudiantes a quienes está dirigido el presente proyecto puedan conocer los diferentes métodos de codificación de Línea mediante la visualización de la forma en que se realiza la codificación de Línea.

Por lo que se requiere tener un buen manejo de este software para que el programa desarrollado cumpla con los requerimientos planteados.

2.4.3 COSTOS DE ELABORACION

Para la obtención del costo del presente proyecto se lo ha divido en varias actividades y a cada una de ellas se le ha asignado un costo con el objetivo de poder

obtener el costo aproximado del equipo para la codificación y decodificación de línea planteado en este proyecto, a continuación se indica un breve detalle de los diferentes componentes que se tomaron en cuenta para obtener el costo del equipo.

El primer rubro que se tomó en cuenta es el del costo del diseño del módulo de codificación y decodificación de línea en el cual se toma en cuenta una aproximación de las horas empleadas para el diseño del mismo, otro rubro es el de los elementos que se emplearan para la construcción del hardware del equipo y por último el de las pruebas para comprobar el perfecto funcionamiento de todos los componentes del módulo, cada uno de estos rubros se los detalla en la Tabla 2.1.

Componentes **Rubros**	**Fuentes de Financiamiento** **Recursos Propios (USD)**	**TOTAL** **(USD)**
1. Diseño	500	500
1.1 Diseño	500	
2. Desarrollo de Hardware	**160**	160
2.1.Compra Elementos Electrónicos	110	
2.2. Compra Elementos Adicionales	50	
3. Implementación	**40**	40
3.1. Pruebas de Funcionamiento	10	
3.2. Acabados del Equipo	30	
	TOTAL	**700**

Tabla 2.1: Costo aproximado del equipo de laboratorio

CAPITULO III

Desarrollo de la Solución del Problema

3.1 INTRODUCCION

En el presente capítulo, se reseñaran las definiciones de los temas relacionados al desarrollo del proyecto además de dar a conocer el procedimiento que se ha realizado, para cumplir con los objetivos específicos planteados.

3.2 MARCO TEORICO

3.2.1 MICROCONTROLADORES PIC

Los Microcontroladores denominados PIC de sus siglas en ingles ***P****eripheral* ***I****nterface* ***C****ontroller* o Controlador de Interfaz Periférico fabricados por la Microchip Technology Inc. son dispositivos del tipo RISC es decir que posee dos características fundamentales:

- Las instrucciones son de tamaño fijo y son presentadas en un reducido número de formatos.
- Únicamente las instrucciones de carga y almacenamiento acceden a la memoria de datos

Los Microcontroladores (Figura 3.1) cuentan con un microprocesador o unidad de procesamiento central, una memoria para almacenar el programa, una memoria para almacenar datos y puertos de entrada y salida, estos dispositivos son unidades autosuficientes y más económicas, el funcionamiento de los Microcontroladores está determinado por el programa en el almacenado el cual puede escribirse en diferentes lenguajes de programación.

Figura 3.1: General Instrument PIC1650
Fuente: **http://es.wikipedia.org**

Por las características mencionadas y su alta flexibilidad, estos dispositivos son utilizados en una gran variedad de sistemas para controlar maquinas, componentes de sistemas complejos como aplicaciones industriales de automatización y robótica, domótica, equipos médicos, etc.

3.2.1.1 Arquitectura Microcontrolador PIC 18FXXX

Los Microcontroladores PIC18FXXX poseen la arquitectura denominada Harvard, con una memoria de datos de 2048 Bytes, 256 bytes de EEPROM, una memoria de programa de 16.384 instrucciones esta arquitectura se la muestra en la Figura 3.2.

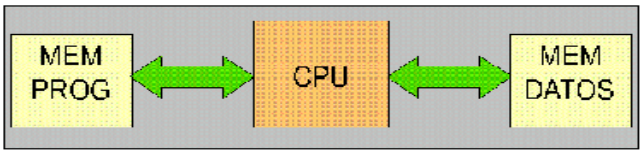


Figura 3.2: Arquitectura Harvard.

La arquitectura Harvard, consiste en un esquema en el cual el CPU está conectado a dos memorias por intermedio de dos buses separados (Figura 3.2), la primera memoria contiene únicamente las instrucciones del programa, y es llamada Memoria de Programa, la segunda memoria es exclusivamente para almacenar los datos y es llamada Memoria de Datos como se puede ver en la Figura 3.2, ambos buses son totalmente independientes y pueden ser de distintos anchos.

Este tipo de arquitectura permite por ejemplo para un procesador del tipo RISC (Reduced Instrucción Set Computer) que el set de instrucciones y el bus de la memoria de programa se puedan diseñar de tal forma que todas las instrucciones tengan una sola posición de memoria de programa de longitud.

Además, como los buses son independientes, el CPU puede estar accediendo a los datos para completar la ejecución de una instrucción, y al mismo tiempo estar leyendo la próxima instrucción a ejecutar, por lo que se puede observar claramente que las principales ventajas de esta arquitectura las cuales son:

- El tamaño de las instrucciones no se encuentra relacionado con el de los datos, y por lo tanto puede ser optimizado para que cualquier instrucción ocupe una sola posición de memoria de programa, con el cual se logra una mayor velocidad y una menor longitud de programa.

- El tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad de operación.

Una desventaja de los procesadores que tienen la arquitectura Harvard, es que deben poseer instrucciones especiales para poder acceder a tablas de valores constantes los cuales pueden ser incluidos en los programas, ya que estas tablas se encontraran físicamente en la memoria de programa (por ejemplo en la EPROM de un microprocesador).

3.2.1.2 El Procesador

Este es el elemento más importante de un microcontrolador este elemento es el que determina sus principales características, tanto a nivel hardware como software, se encarga de direccionar la memoria de instrucciones, recibir la instrucción en su curso, su descodificación y la ejecución de la operación que implica dicha instrucción, así como la búsqueda de los operandos y el almacenamiento del resultado, existen tres orientaciones en cuanto a la arquitectura o funcionalidad de los procesadores actuales.

CISC.- Un gran número de procesadores usados en los Microcontroladores están basados en la filosofía CISC o Computadores de Juego de Instrucciones Complejo los cuales disponen de más de 80 instrucciones de máquina alguna de las cuales son muy sofisticadas y potentes, por lo que requieren varios ciclos para una ejecución.

RISC.- Esta clase de procesadores del tipo RISC o Computadores de Juego de Instrucciones Reducido, tienen instrucciones de máquina muy reducidas, estas instrucciones son simples y generalmente se ejecutan en un ciclo de máquina. La sencillez y rapidez de las instrucciones permiten optimizar el hardware y el software del procesador.

SISC.- De sus siglas Computadores de Juego de Instrucciones Específico, estos procesadores con utilizados en los Microcontroladores destinados a aplicaciones muy concretas, el juego de instrucciones, además de ser muy reducido, es "específico", es decir, las instrucciones se adaptan a las necesidades de la aplicación.

3.2.1.3 MICROCONTROLADOR PIC 18F4550

Los Microcontroladores PIC18F4550 presentan las siguientes características básicas:

- Poseen la arquitectura RISC avanzada Harvard: 16 bits con 8 bits de datos.
- Tienen 77 instrucciones con 83 extensiones.
- Existen encapsulados desde 18 hasta 80 pines.
- Hasta 64Kbytes de memoria de programa (hasta 2 Mbyte en ROMless).
- Multiplicador Hardware 8x8.
- Frecuencia máxima de reloj 40MHz, hasta 10 MIPS.
- Pila de 32 niveles, múltiples fuentes de interrupción.
- Periféricos de comunicación avanzada (CAN y USB), entre otros.
- 32K de memoria flash (16384 palabras).
- Divisor PLL para frecuencia de clock.
- Puerto SPI.
- Módulos de captura de 16 bits de resolución.
- Posee dos salidas de PWM.
- Transceptor USB con regulador de voltaje integrado.
- Memoria RAM de doble acceso por USB de 1Kbyte.
- Soporta transferencia por control, interrupción, asíncrona y Bulk.
- Velocidad baja (1.5 Mb/s) y alta (12 Mb/s).
- Compatible con USB V 2.0.

Este microcontrolador incluye también otras características como son: 2 Kbyte de RAM, de los cuales 1 Kbyte puede ser dedicado al buffer USB, 256 bytes de memoria de datos EEPROM, módulo EUSART que soporta interfaz serie RS232, RS485 y LIN, puerto SSP de comunicaciones serie I2C™ y SPI™, convertidor A/D de 10 bits de resolución con más de 12 canales, 2 comparadores analógicos, módulo

Captura/Comparación/PWM con 16 bits de resolución, 4 temporizadores (3 de 16 bits y uno de 8 bits), circuito detector de bajo voltaje, reset por caída de tensión programable y mejoras en la depuración en circuito con tres puntos de ruptura. Una de las consideraciones claves para los diseñadores de aplicaciones USB es la disponibilidad de soporte firmware.

La distribución de pines que posee el Microcontrolador PIC 18F4550 la podemos observar en la Figura 3.3. Un mayor detalle de las características de este dispositivo se las puede encontrar en el Anexo 2.

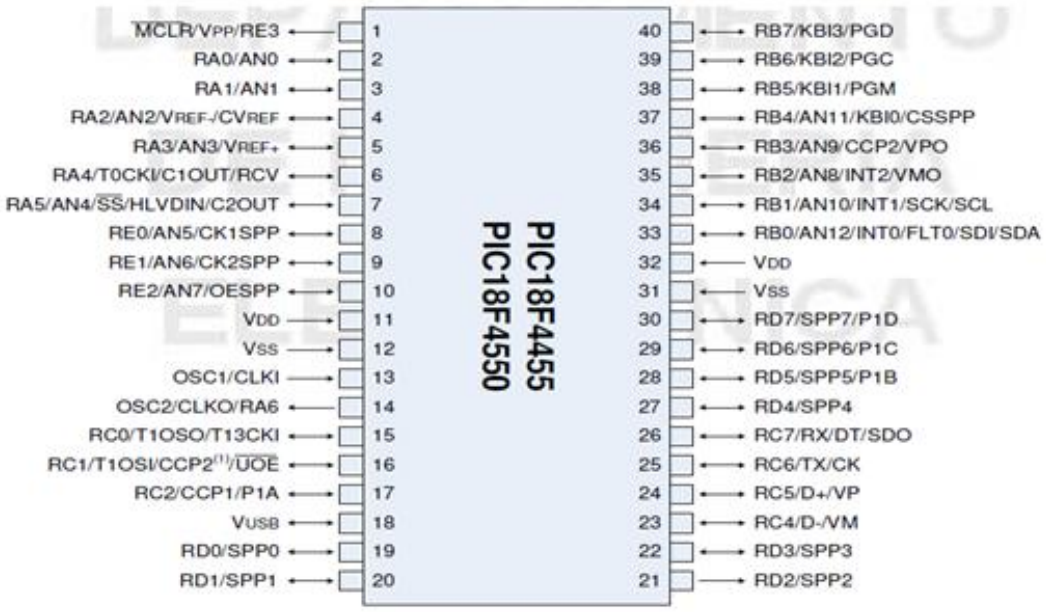


Figura 3.3: Distribución de pines PIC 18F4550

Fuente: **http://ww1.microchip.com**

3.2.2 SOFTWARE LABVIEW

LabView es un entorno de programación el cual es utilizado para desarrollar sistemas de medida, pruebas y control mediante el uso del lenguaje de programación G o Gráfico. Ofrece una integración con miles de dispositivos de hardware, tiene cientos bibliotecas integradas para el análisis avanzado y visualización de datos, con el fin de crear instrumentación virtual, un ejemplo de este tipo de programación se muestra en la Figura3.5 que corresponde al programa final de una comunicación RS-232 PC-PIC-PC con LabView.

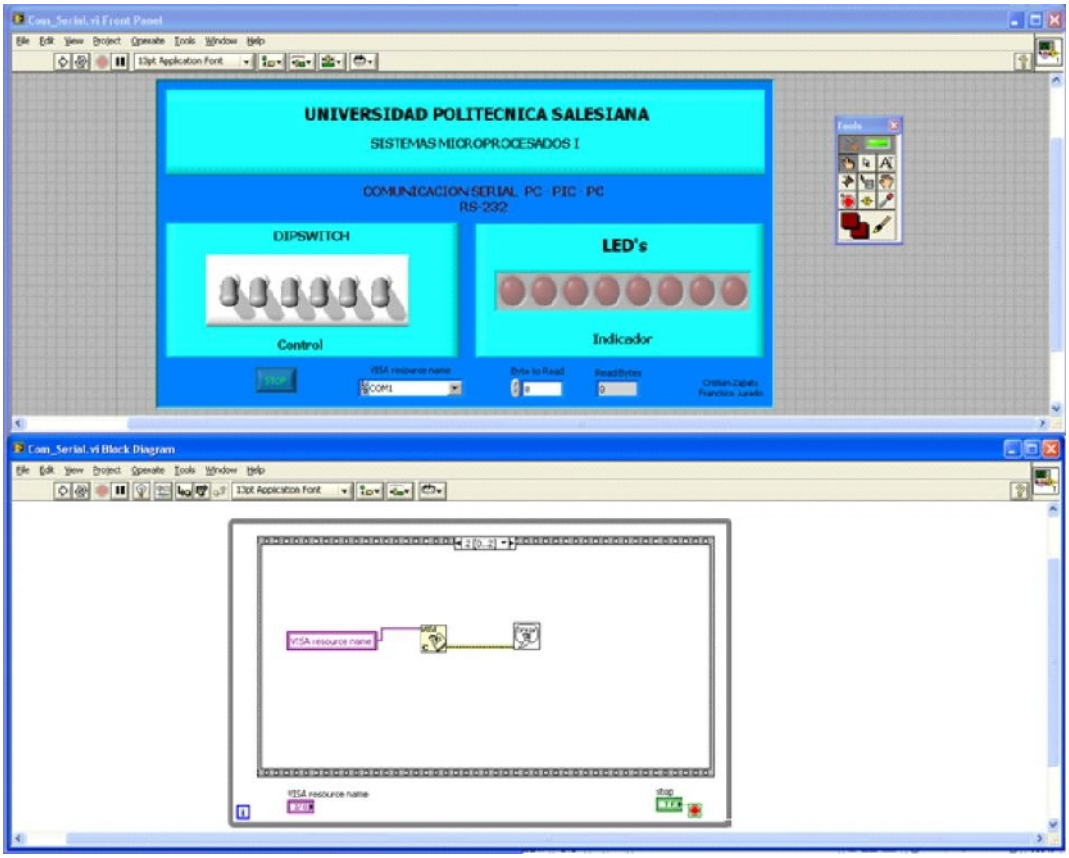


Figura 3.4: Entorno Gráfico LabView

3.2.2.1 Principales Características

La principal característica de este software es su facilidad de uso, los programas en LabView son llamados instrumentos virtuales o VIs, estos VIs son interactivos ya que poseen una configuración que permite al usuario personalizar su funcionalidad además de que presenta facilidades para el manejo de:

- Interfaces de Comunicaciones como Puerto Serie, Puerto Paralelo, GPIB, PXI, VXI, TCP/IP, UDP, DataSocket, Irda, Bluethooth, USB, OPC.
- Capacidad de interactuar con otros lenguajes y aplicaciones como librerías de funciones DLL, .NET, ActiveX, Multisim, SolidWorks, etc.
- Herramientas gráficas y textuales para el procesado digital de señales.
- Visualización y manejo de gráficas con datos dinámicos.
- Adquisición y tratamiento de imágenes.
- Control de movimiento.

- Tiempo Real.
- Programación de FPGAs para control o validación.
- Sincronización entre dispositivos.

Al ser una herramienta gráfica de programación, los programas son gráficos lo que facilita su compresión y al poseer ya prediseñados una gran cantidad de bloques facilitan al usuario la creación del proyecto, cada VI consta de dos partes diferenciadas El Panel Frontal (Front Panel) y el Diagrama de Bloques (Block Diagram).

3.2.2.2 Panel Frontal

Es la parte del programa que sirve para interactuar con el usuario cuando el programa se está ejecutando, el o los usuarios podrán observar los datos del programa actualizados en tiempo real. En esta interfaz se definen los controles e indicadores, en la Figura 3.5 se muestra el Panel Frontal de un programa de comunicación USB entre PC-PIC-PC el cual enciende los diodos leds conectados al PIC desde la PC y viceversa encender los indicadores desde los interruptores conectados al PIC.

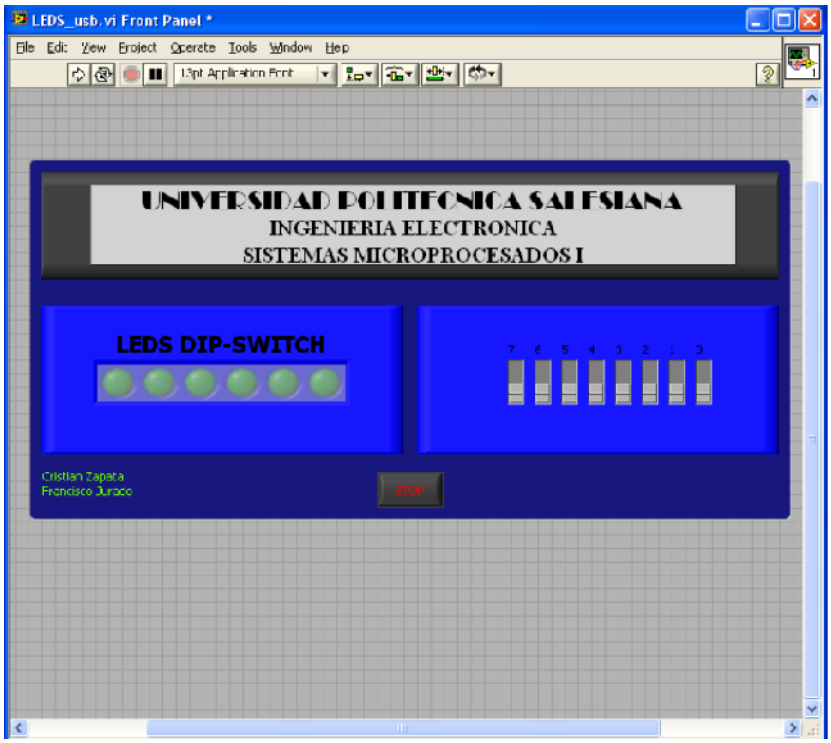


Figura 3.5: Panel Frontal Comunicación PC-PIC-PC por medio del puerto USB

3.2.2.3 Diagrama de Bloques

Este constituye el código fuente del VI, es aquí en donde se realiza la implementación del programa para controlar o realizar cualquier procesamiento de las entradas y salidas que fueron creadas en el Panel Frontal, este incluye funciones y estructuras integradas en las librerías que incorpora LabView, en la Figura 3.6 se muestra el Diagrama de Bloques de un programa de comunicación USB entre PC-PIC-PC.

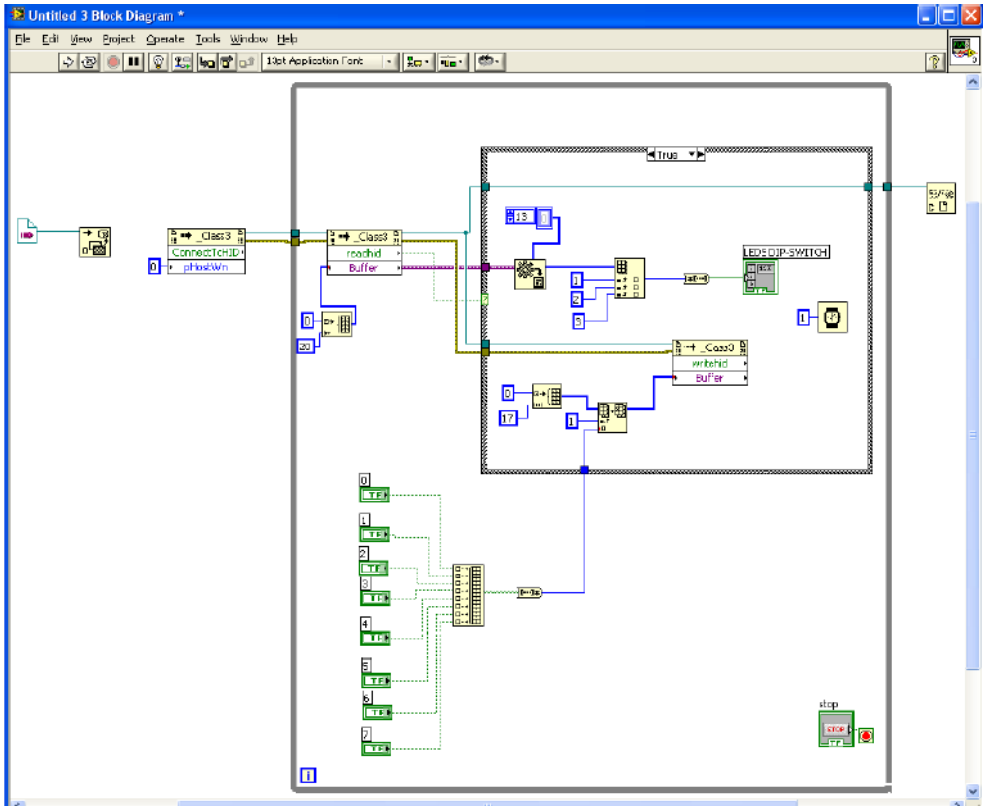


Figura 3.6: Diagrama de Bloques Comunicación PC-PIC-PC por medio del puerto USB

3.2.3 CODIGOS DE LINEA

Los códigos de línea son utilizados para el transporte digital de datos, consisten en representar la amplitud de la señal digital que va a ser transportada con respecto al tiempo, la representación de la onda se suele realizar mediante un número determinado de impulsos, estos impulsos representan los 1s y 0s digitales.

Existen diversos tipos de codificación en banda base, los cuales se los puede dividir de acuerdo a las características que determinan su espectro de señal en:

- **Unipolar:** Se lo denomina así porque en la señal todos los niveles de amplitud se encuentran a un mismo lado del eje del tiempo, en otras palabras, todos los niveles tienen valores de voltajes mayores o iguales a cero, o todos los valores son menores o iguales a cero (Figura 3.7).

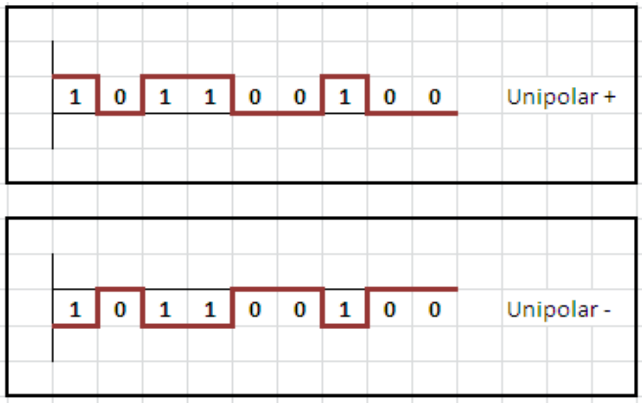


Figura 3.7: Ejemplo de Código Unipolar

- **Bipolar:** Se lo denomina así porque en la señal los niveles de amplitud se los pueden encontrar en ambos lados del eje de tiempo, en otras palabras, pueden existir valores positivos o negativos indistintamente.

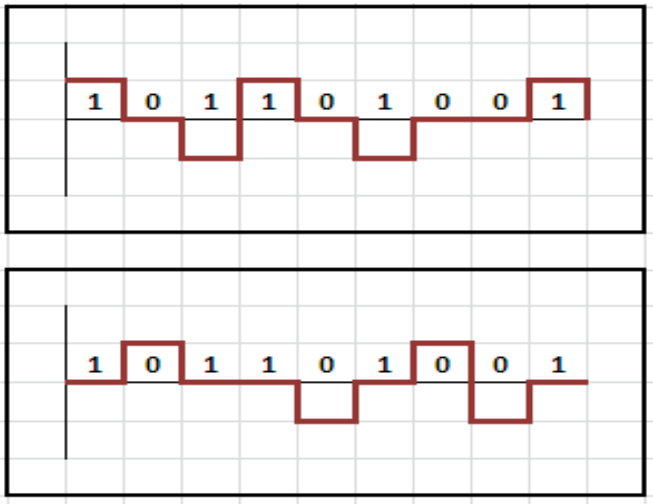


Figura 3 8: Ejemplo de Código Bipolar

- **Multinivel:** se la denomina así porque se pueden tener diferentes tipos de elementos de señal, obteniendo diferentes niveles de amplitud de señal para codificar los elementos de datos, permitiendo un aumento en la velocidad de transmisión, aumentando el número de bits por baudio que se codifican por elemento de señal.

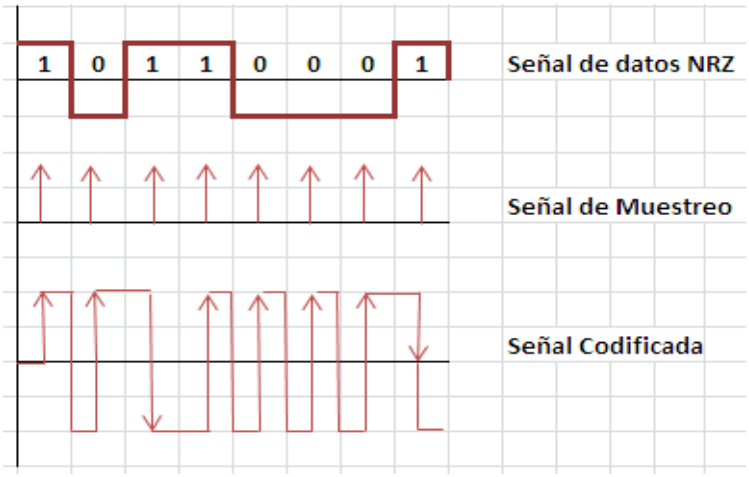


Figura 3.9: Ejemplo de Código Multinivel

- **Polar:** se la conoce así porque la señal asume un valor positivo (+V) para un digito binario y otro valor (-V) para otro digito binario, la señal que se obtiene como resultado también conocida como onda cuadrada al igual que la señal Unipolar admite lógica directa es decir cuando el bit 1 asume el valor de +V el bit 0 el valor de –V e inversa cuando el bit 0 y el bit 1 toman el valor de +V y –V respectivamente, adicionalmente se puede obtener una modificación de esta señal cuando esta tiene un retorno a cero.

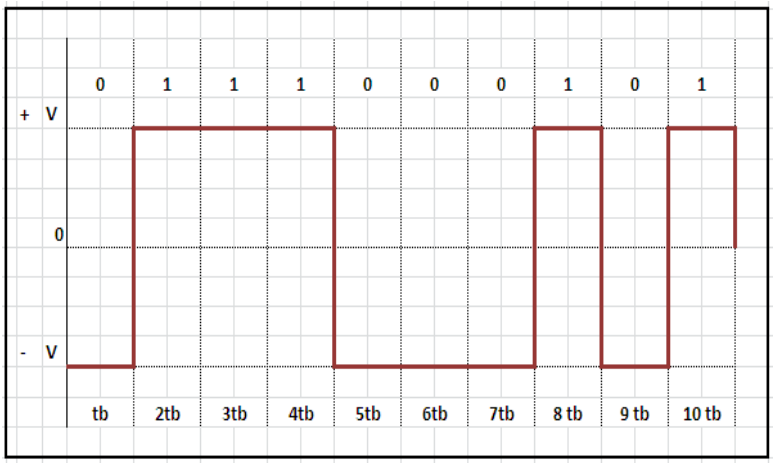


Figura 3.10: Señal polar con retorno a cero con lógica directa.

El objetivo de aplicar un código a una señal de banda base es obtener un código de línea con el propósito de evitar los efectos indeseables de aquellas componentes continuas, componentes de baja frecuencia y las dificultades que se presentan al reconstruir la señal de reloj.

Los códigos le línea para efecto de estudio parten de un algoritmo que se aplica sobre una señal de banda base, ya que estrictamente no es de esta forma en la que funcionan un transceptor, este equipo para determinar la forma de la onda que se da en una cadena de bits traza una señal en banda base y la codifica con el objetivo de generar una nueva onda.

Los códigos de línea tienen la finalidad de mejorar las prestaciones de los sistemas de transmisión, el esquema de codificación o la asignación de bits de datos a elementos de señalización simplemente es la correspondencia que se establece entre los bits de datos y los elementos de la señal, para poder conseguir que la información transmitida llegue al receptor con la máxima fidelidad o con la mínima distorsión posible un código de línea debe poseer las siguientes características:

- *Eliminación de componente continúa en la señal transmitida*, esto permite el acoplamiento AC, ya que algunos de los sistemas que utilizan el acoplamiento por transformador tienen poca sensibilidad a las bajas frecuencias.

- *Facilidad para la extracción de reloj*, debido a que el receptor necesita tener una copia del reloj emisor para poder decodificar la señal recibida.

- *Detección de errores*, debido a que algunos sistemas poseen la propiedad de detectar errores sin necesidad de incrementar el número de bits en el código primario.

- *Eficiencia espectral*, con la finalidad de aprovechar el ancho de banda para el envió de datos con la misma ocupación de un ancho de banda determinado.

- *Inmunidad al ruido,* debido a que algunos códigos son más susceptibles de sufrir afectaciones debido al ruido que otros, para este caso los códigos bipolares se comportan mejor que los unipolares.

3.2.3.1 Terminología Básica

Antes de continuar con el estudio de los códigos de línea más utilizados en los sistemas de comunicaciones, se realiza una breve revisión de algunos conceptos fundamentales que ayudaran al entendimiento del tema.

Ancho de Banda (BW): Este representa el rango de frecuencias en donde se concentra la mayor parte de la potencia de la señal que se está usando, su unidad de medida son los Hertz (Hz) y se lo puede calcular de forma matemática mediante la utilización de la transformada de Fourier.

$$BW=\Delta f=fcs-fci \qquad \text{(Ec 3.1)}$$

Siendo fcs y fci las frecuencias de corte Superior e Inferior, respectivamente.

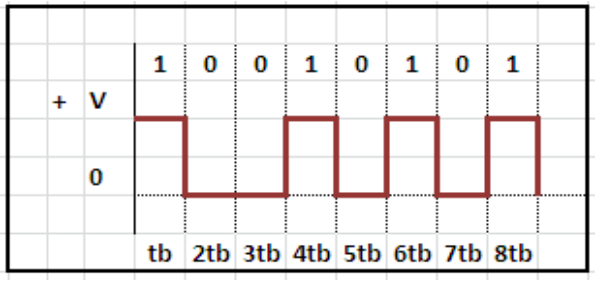
Tiempo de bit (Tb): Se lo define como el tiempo que se requiere para que un bit pueda colocarse y transmitirse en un sistema determinado.

Intervalo de impulso (τ): Es la duración mínima del impulso usado para representar la información.

Velocidad de Transmisión (Vt): Se define como el número de bits transmitidos por unidad de tiempo. Matemáticamente viene expresado como:

$$Vt = \frac{\#\text{bits transmitidos}}{T} \text{ (bps)} \qquad \text{(Ec. 3.2)}$$

En la Figura 3.11 se puede observar una cadena de Bits en un periodo de tiempo determinado y el cálculo de la velocidad de transmisión que se alcanza en este intervalo de tiempo.



$$\text{Vt} = \frac{8\ bits}{1\ seg}$$

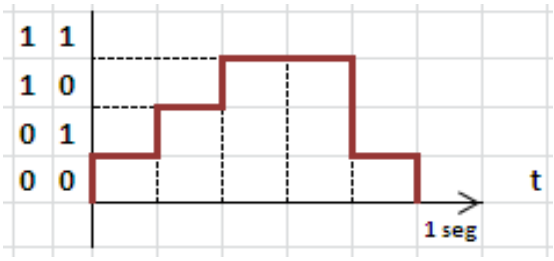
$$\text{Vt} = 8\ (bps\ )$$

Figura 3.11: Velocidad de transmisión

Velocidad de Señalización o Modulación (Vs): Se define como el número de símbolos o estados transmitidos por unidad de tiempo. Matemáticamente viene expresado como:

$$\text{Vs} = \frac{\#\text{estados transmitidos}}{\text{T}} \quad \text{(Baudios) (Ec. 3.3)}$$

En la Figura 3.12 se puede observar los diferentes estados que pueden tener una señal en un periodo de tiempo, y el cálculo de la velocidad de señalización para este caso.



$$\text{Vt} = \frac{5\ estados}{1\ seg}$$

$$\text{Vt} = 5\ (baudios\ )$$

Figura 3.12: Velocidad de señalización

Tasa de Bits (R): Número de bits transmitidos por segundo. Su unidad son los bps (bits por segundo) y se representa matemáticamente como:

$$R = \frac{1}{\text{Tb}}\ (bps) \qquad \text{(Ec. 3.4)}$$

Marca y **espacio**: Se los conoce como los dígitos binarios 1 y 0, respectivamente.

3.2.3.2 Criterios para la Codificación de Datos

Para realizar la codificación de datos se debe tener presente que:

- Un **aumento** de la velocidad de transmisión de los datos (Vt) provoca un **aumento** de la tasa de error de bits (BER).
- Un **aumento** de la relación señal/ruido (SNR) provocara una **disminución** de la tasa de error BER.
- Un **aumento** en el ancho de banda (BW) permite **aumentar** la velocidad de transmisión de datos (Vt).
- Otro factor que permite aumentar el rendimiento es el **esquema de codificación**, dentro de los esquemas de codificación que existen podemos citar a los siguientes:
  - No retorno a cero – nivel (NRZ-L).
  - No retorno a cero – invertido (NRZ-I).
  - Bipolar AMI.
  - Pseudoternario.
  - Manchester.
  - Manchester diferencial.
  - B8ZS.
  - HDB3

3.2.3.3 Tipos de Códigos de Línea

3.2.3.3.1 *Código de Línea NRZ*

Se denomina NRZ de sus siglas en ingles Non Return to Zero o Sin retorno a cero, este código es el que permite de forma frecuente y fácil la transmisión de señales digitales mediante el uso de un nivel diferente de tensión para cada uno de los bits, este código no es autosincronizante, su principal ventaja es que al emplear pulsos de

larga duración requiere menor ancho de banda que otros sistemas de codificación que emplean pulsos más cortos.

Estos códigos mantienen el nivel de tensión constante durante el tiempo de bit (Tb), es decir que durante el Tb no existen transiciones o no hay retorno al nivel cero de tensión entre bits consecutivos, mediante la asignación de un nivel de voltaje a cada símbolo se facilita la decodificación de un mensaje, esta es básicamente la forma en que funciona el código NRZ, dependiendo de la forma de onda binaria se puede clasificar a este código en NRZ Unipolar y NRZ polar.

Características Generales

Los códigos del tipo NRZ presentan las siguientes características generales:

- Fáciles de implementar.
- Presentan un uso eficaz del ancho de banda.
- Presentan una gran capacidad de sincronización.
- Capacidad de detección de errores.

Problemas Existentes

El código NRZ presenta un problema en la longitud de la secuencia de unos y ceros debido a que el receptor necesita sincronizarse, comprobar que existe señal o que no se encuentra disponible.

En el caso de tener una prolongada permanencia en el nivel de la señal positivo o negativo durante la transmisión puede producir el denominado desplazamiento de la línea base, ocasionando al receptor la dificultad de realizar una adecuada decodificación.

Otro aspecto negativo se presenta en el método que se debe emplear para que el emisor y el receptor se sincronicen, siendo para ello necesario continuos cambios en la señal lo que se dificulta cuando existen cadenas de unos y ceros que mantienen niveles altos y bajos de tensión por largos períodos de tiempo.

Los códigos NRZ a pesar de presentar un uso eficaz del ancho de banda, este no resulta ser eficiente.

Esta clase de códigos son muy susceptibles a interferencias.

Cuando se transmiten secuencias largas de unos y ceros, los límites entre bits individuales se pueden perder.

Aplicaciones

La principal aplicación que se da a los códigos de línea NRZ es la grabación magnética, pero presentan una gran limitación para la transmisión de señales.

3.2.3.3.1.1 *Código de Línea NRZ Unipolar.*

Este código presenta la característica de que el voltaje que representa los bits varía entre cero y un valor positivo o negativo de voltaje, este tipo de código no es recomendable para transmisiones a largas distancias debido principalmente a dos motivos, el primero debido a que presentan niveles residuales de corriente continúa y el segundo debido a que existe la posible ausencia de un suficiente número de transiciones de la señal que permitan la recuperación fiable de una señal de temporización.

Este código consiste básicamente en asignar un valor de señal de +V o -V al símbolo "1" mientras que al símbolo "0" se le asignara el valor de 0V, se puede trabajar también con lógica inversa dependiendo del caso es decir se asignara un valor de señal de +V o -V al símbolo "0" mientras que al símbolo "1" se le asignara el valor de 0V, en la Figura 3.13 se observa la gráfica que muestra la representación de los bits codificados con el código NRZu.

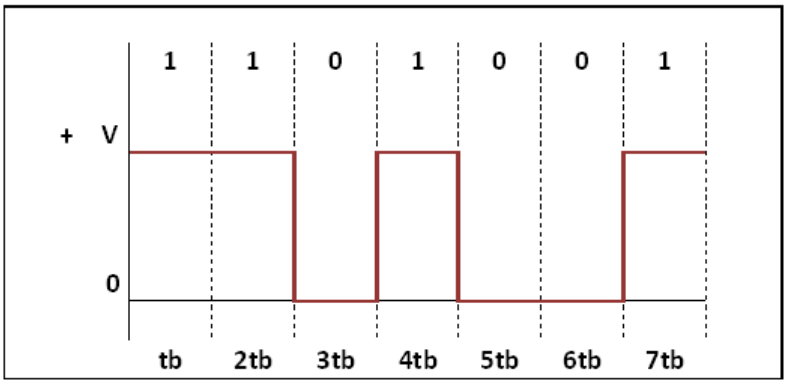


Figura 3.13: Representación de bits usando el código NRZu lógica positiva

3.2.3.3.1.2 *Código de Línea NRZ Polar*

Este código a diferencia del anterior desplaza el nivel de referencia de la señal reduciendo a la mitad la diferencia de potencial necesaria con referencia a la Unipolar, utiliza dos niveles de voltaje +V y –V para representar a un bit sea este cero o uno.

Este código asigna un valor de señal de +V al símbolo "1" y al símbolo "0" se le asignara el valor de -V, o también se puede asignar un valor de señal de +V al símbolo "0" mientras que al símbolo "1" se le asignara el valor de –V, la representación de los bits con el código NRZp se muestra en la Figura 3.14.

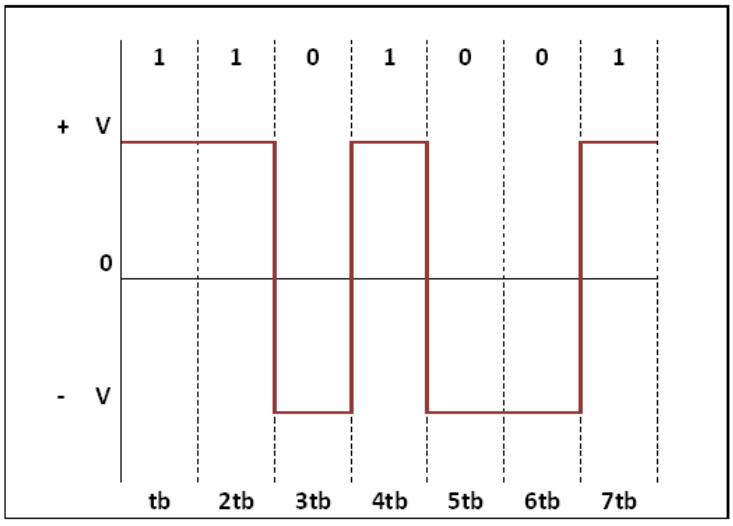


Figura 3.14: Representación de bits utilizando el código NRZp.

3.2.3.3.2 *Código de Línea AMI*

El código AMI de sus siglas en inglés Alternate Mark Inversión o Inversión de marcas alternadas es un código de línea que se lo utiliza para transmisiones binarias, se lo puede definir como un código bipolar con Retorno a cero que posee características especiales, en este código, cuando se asigna un impulso positivo al primer “1”, al siguiente "1" se le asigna un impulso negativo, y así sucesivamente asignando alternadamente impulsos positivos y negativos a los "1" lógicos.
Este tipo de esquema de codificación ofrece la ventaja de que la sincronización es mucho más sencilla, y únicamente la aparición de largas cadenas de ceros dificulta la misma, además no permite la aparición de componentes continuas debido a la alternancia de los pulsos, dependiendo de la forma de onda binaria se puede clasificar en código AMI Normal y código AMI Invertido.

Características Generales.

Para este código el espectro de la señal a la frecuencia de valor cero debe ser cero, para eliminar en la mayoría de los canales la componente continua de las señales.

El máximo espectral debe darse en un submúltiplo o en la proximidad de un submúltiplo de régimen binario, de esta forma la energía que se necesita para producir la señal estará en la zona en la que la atenuación de transmisión del cable es más reducida y la atenuación de la diafonía es mayor, logrando así conseguir una mejor relación señal ruido.

Se reducen los requerimientos de potencia para la transmisión y se logra una mayor inmunidad a la diafonía.

Las condiciones anteriores permiten que si la señal contiene largas secuencias de ceros se utiliza un aleatorizador que limite estadísticamente el número de ceros consecutivos, es decir que este código se encarga de poner un 1 lógico saltando un 0 lógico en la parte de transmisión mientras que en la recepción lo invierte, caso contrario se perdería el sincronismo con el reloj. Además, si se encuentran dos unos seguidos con la misma polaridad sabemos que se ha producido un error.

Aplicaciones

El código AMI fue utilizado extensamente en la primera generación de redes PCM, y aún se suele ver en los multiplexadores más antiguos, su éxito radica en que no hay un gran número seguido de ceros en su código, de esta forma se logra obtener no más de 15 ceros consecutivos, lo que asegura la sincronización.

La forma de este código se aplica en los sistemas troncales T1 a una velocidad máxima de 1.544 Mbps, y en la transmisión de canales B. en la Red Digital de Servicios Integrados (RDSI) de acuerdo con la Recomendación UIT-T I.430.

3.2.3.3.2.1 *Código AMI Normal*

Este código presenta dos variantes el AMI Normal del tipo NRZ y el AMI Normal del tipo RZ.

3.2.3.3.2.1.1 *Código AMI Normal del tipo NRZ*

En este código se representa al símbolo "1" con valores de voltaje positivos y negativos de forma alternada mientras que para el símbolo "0" tomara el valor de 0V, tal como se muestra en la Figura 3.15.

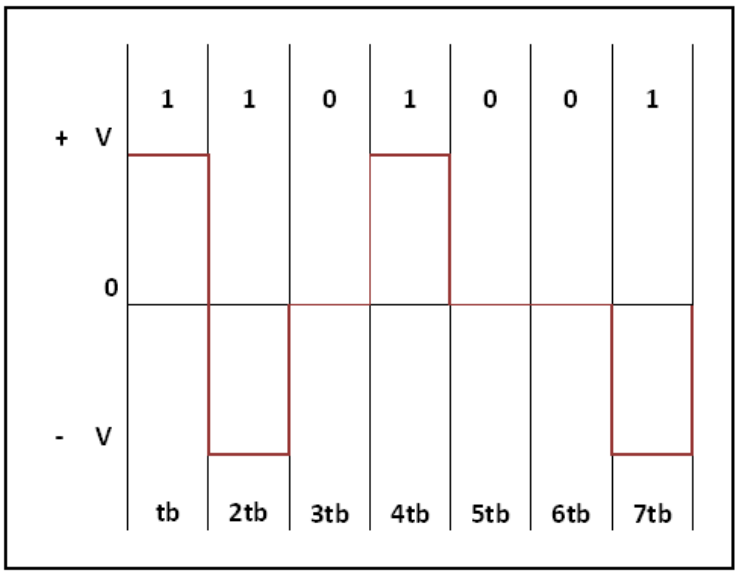


Figura 3.15: Representación de bits utilizando el código AMI Normal NRZ

3.2.3.3.2.1.2 *Código AMI Normal del tipo RZ*

En este código se representa al símbolo “1” con valores de voltaje positivos y negativos de forma alternada que a diferencia del anterior en la mitad del tiempo de bit Tb de este símbolo se produce una transición por cero es decir que este símbolo tomara valores equivalentes a $\frac{Tb}{2}$ para valores positivos y $-\frac{Tb}{2}$ para valores negativos, mientras que para el símbolo “0” tomara el valor de 0V, tal como se muestra en la Figura 3.16.

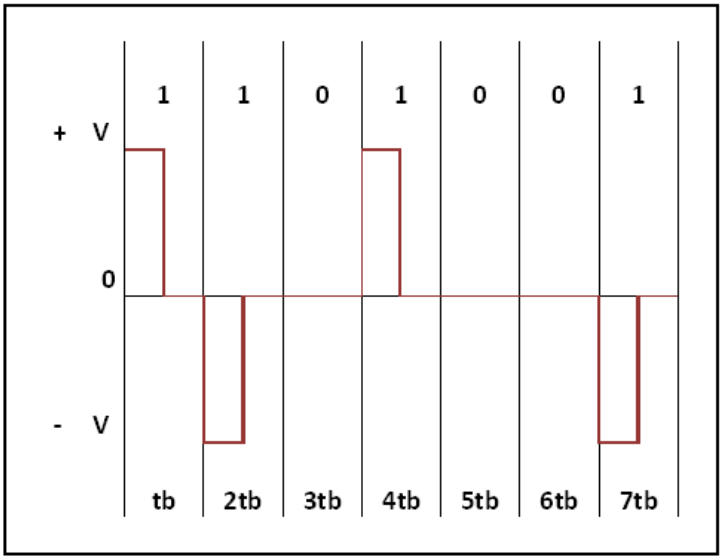


Figura 3.16: Representación de bits utilizando el código AMI Normal RZ

3.2.3.3.2.2 *Código AMI Inverso*

Este código al igual que el AMI Normal presenta dos variantes el AMI Inverso del tipo NRZ y el AMI Inverso del tipo RZ.

3.2.3.3.2.2.1 *Código AMI Inverso del tipo NRZ*

En este código utiliza la lógica inversa, es decir, se representa al símbolo “0” con valores de voltaje positivos y negativos de forma alternada mientras que para el símbolo “1” tomara el valor de 0V, tal como se muestra en la Figura 3.17.

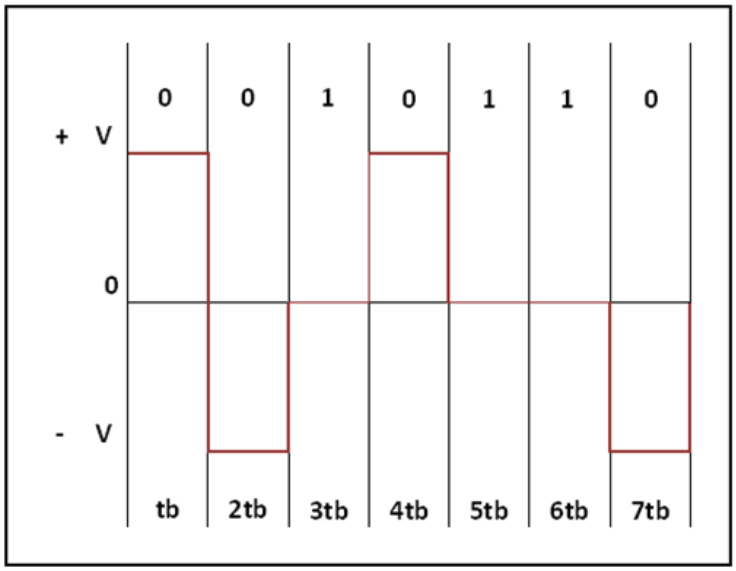


Figura 3.17: Representación de bits utilizando el código AMI Inverso NRZ

3.2.3.3.2.2.2 *Código AMI Inverso del tipo RZ*

En este código se representa al símbolo "0" con valores de voltaje positivos y negativos de forma alternada que a diferencia del anterior en la mitad del tiempo de bit Tb de este símbolo se produce una transición por cero, es decir, que este símbolo tomara valores equivalentes a $\frac{Tb}{2}$ para valores positivos y $-\frac{Tb}{2}$ para valores negativos, mientras que para el símbolo "1" tomara el valor de 0V, tal como se muestra en la Figura 3.18.

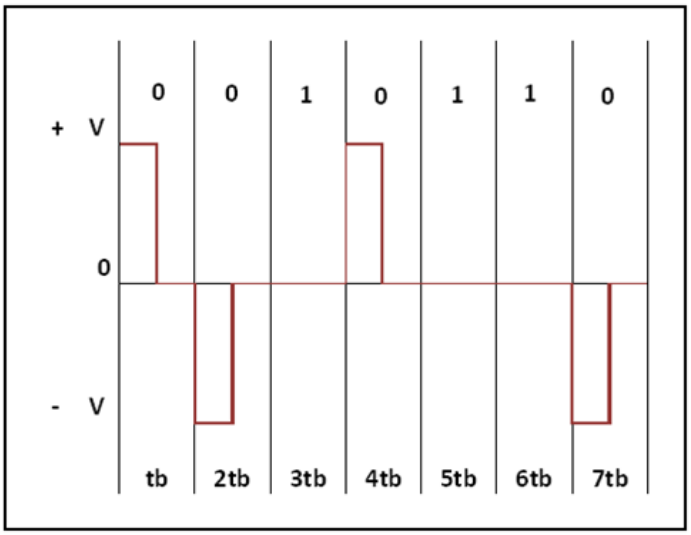


Figura 3.18: Representación de bits utilizando el código AMI Inverso RZ

3.2.3.3.3 *Código de Línea Diferencial NRZ*

La codificación diferencial tiene la característica de que el valor de la señal de un elemento se codifica en base a la polaridad del elemento de señal anterior en vez de codificarlo como valor absoluto de la señal, existiendo dos variantes denominadas diferencial NRZ tipo marca y diferencial NRZ tipo espacio.

Esta clase de códigos son utilizados en la mayoría de los casos para transmisiones de baja velocidad del orden de los 100 Kbps, este tipo de codificación permite romper una cadena de dígitos binarios 1 sin la necesidad de usar el retorno a cero

Características Generales

Los códigos diferenciales presentan una componente continua tenue o que tiende al valor de cero como toda señal polar.

En este esquema de codificación es más confiable detectar una transición en presencia de ruido que comparar a un valor de un nivel.

En un sistema de transmisión complejo es fácil perder el sentido de polaridad de la señal. Por ejemplo, sobre una línea de multicaída, si los pares trenzados que conectan a un dispositivo se invierten, los **unos** y **ceros** son invertidos, esto no acontece con la codificación diferencial.

Estos códigos son fáciles de implementar y además hacen un uso eficiente del ancho de banda pero no presentan solución al problema generado cuando existe una cadena larga de dígitos binarios 0.

Aplicaciones

El empleo de esta clase de códigos se limita en general para transmisiones asíncronas.

3.2.3.3.3.1 *Código Diferencial NRZ tipo marca*

También conocido como NRZ – I o NRZ Invertido es utilizado por el protocolo SDLC de la IBM, para este código la existencia de un digito binario de valor "1" nos indica que existirá un cambio del estado anterior tomando valores de tensión de +V o –V, mientras que la existencia de un digito binario "0" no se producirá transición alguna en otras palabras mantendrá el estado anterior tal como se muestra en la Figura 3.19.

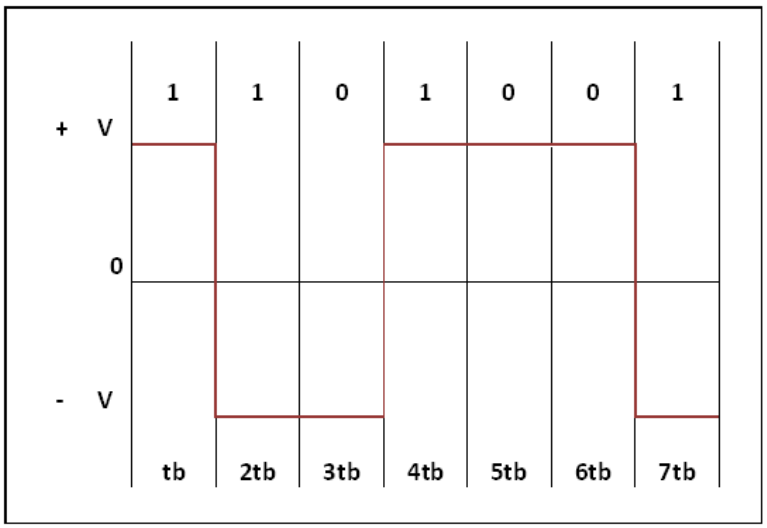


Figura 3 19: Representación de bits usando el código Diferencial NRZ-M

3.2.3.3.3.2 *Código Diferencial NRZ tipo Espacio*

Para el código NRZ-S la existencia de un digito binario de valor "0" indica que existirá un cambio del estado anterior tomando valores de tensión de +V o -V, mientras que la existencia de un digito binario "1" no se producirá transición alguna mantendrá el estado anterior como se muestra en la Figura 3.20.

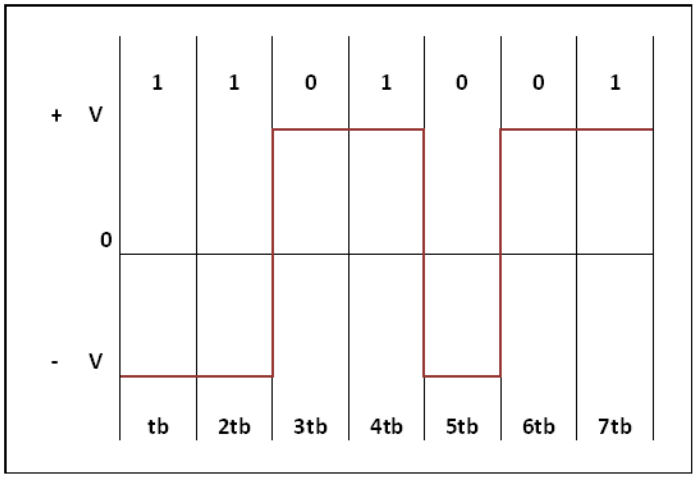


Figura 3.20: Representación de bits usando el código Diferencial NRZ-S

3.2.3.3.4 *Código de línea Manchester*

Este tipo de código es un mecanismo de codificación bipolar, esto quiere decir que a cada digito binario ya sea 0 o 1 se les asocia transiciones de nivel entre dos valores de tensión.

El nivel de tensión se mantiene en un valor durante la mitad del tiempo de bit y tomará otro valor de tensión en la otra mitad, esta transición que se produce a la mitad del Tb permite ser utilizada como elemento de sincronización entre el transmisor y el receptor, en la Figura 3.21, se muestra un ejemplo de la codificación de bits utilizando el código Manchester.

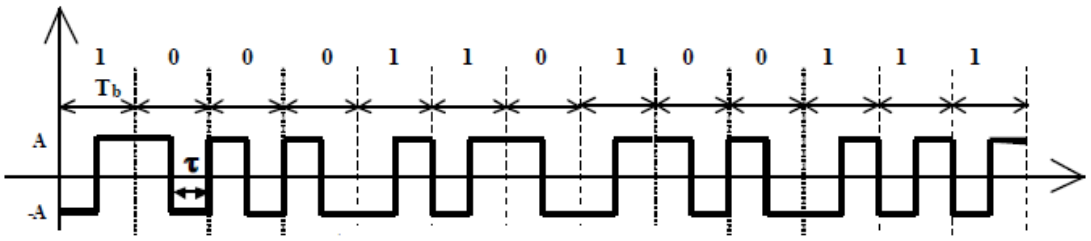


Figura 3.21: Ejemplo de Codificación Manchester

Esta clase de códigos se los creo para mejorar las dificultades que presentan los códigos NRZ, se tiene dos clases el Código Manchester tipo L y el código Manchester Diferencial.

Características Generales

Es posible incorporar mecanismos para la detección de errores los cuales consistirían en comprobar la presencia o la ausencia de las transiciones a la mitad del tiempo de bit ya que en el caso de no existir se estaría produciendo un error.

Es un código de media y alta velocidad que posee la ventaja de que a diferencia de otros códigos no utiliza el valor muestreado en el tiempo de bit para representar un digito binario ya sea este cero o uno sino que utiliza un flanco creciente o decreciente de un pulso para indicarlo.

La sincronización se da debido a la transición que siempre ocurre durante el intervalo de duración correspondiente a un bit, debido a esta característica, se los denomina también auto-sincronizados.

La codificación Manchester no posee componente continua.

Aplicaciones

Se lo suele emplear para la codificación de señales que se transmiten dentro de una red LAN las cuales posean una topología en bus, de par trenzado o de cable coaxial con la ayuda del método de acceso al medio CSMA/CD, es decir con las redes del tipo IEEE.802.3.

3.2.3.3.4.1 *Código Manchester tipo L*

La codificación Manchester tipo L, es un método de codificación eléctrica de una señal binaria es una codificación autosincronizada, ya que de cada bit se obtiene la señal de reloj, lo que hace posible una sincronización precisa del flujo de datos. Una desventaja que presenta este código es el consumo del doble de ancho de banda que una transmisión asíncrona, en la actualidad existen numerosas codificaciones que pueden lograr el mismo resultado consumiendo menor ancho de banda.

La codificación Manchester tipo L se la utiliza en muchos estándares de telecomunicaciones, como por ejemplo Ethernet.

Descripción

Las señales de datos y de reloj, están combinadas en una sola de modo que se autosincroniza el flujo de datos.

Cada bit que se codifica contiene una transición a la mitad del tiempo de bit.

Una transición de negativo a positivo representara un 0 y una transición de positivo a negativo representara un 1, tal como se muestra en la Figura 3.22.

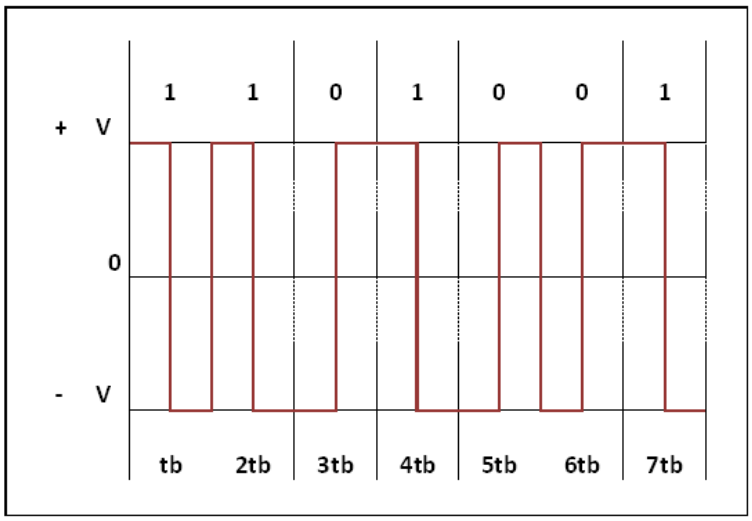


Figura 3.22: Representación de bits utilizando en Código Manchester tipo L

En el código Manchester tipo L cuando se tienen bits iguales y consecutivos se produce una transición al inicio del segundo bit esta transición no es tomada en cuenta por el receptor al momento de decodificar los datos ya que solo toma en cuenta las transiciones separadas uniformemente en el tiempo.

Existen algunas transiciones que no ocurren a mitad de bit, estas transiciones no llevan información útil, y son utilizadas únicamente para colocar la señal en el siguiente estado donde se llevará a cabo la siguiente transición esto permite que la señal se auto-sincronice pero lo que en realidad sucede es que se dobla el

requerimiento de ancho de banda en comparación con otros códigos como por ejemplo los Códigos NRZ.

3.2.3.3.4.2 *Código Manchester Diferencial*

El código Manchester Diferencial es una variación del Manchester tipo L debido a que en ella, un bit de valor 1 se indica por la ausencia de transición al inicio del intervalo, mientras que un bit 0 tiene una transición en el inicio, existiendo siempre una transición en el centro del intervalo.

Este esquema requiere de un equipo más sofisticado que ofrece una mayor inmunidad al ruido. El código Manchester Diferencial presenta como ventajas adicionales las derivadas de la utilización de una aproximación diferencial.

Todas las técnicas bifase fuerzan al menos una transición por cada bit pudiendo tener hasta dos en ese mismos periodo lo que permite tener la máxima velocidad de modulación que resultara el doble que la de los NRZ, esto significa que el ancho de banda necesario es mayor.

Descripción

En el código Manchester Diferencial un bit '1' se indica haciendo que la primera mitad de la señal sea igual a la última mitad del bit anterior, es decir no existe transición al inicio del intervalo mientras que un bit '0' se lo indica haciendo que en la primera mitad de la señal sea contraria a la última mitad del último bit, es decir, con una transición al principio del bit, manteniendo siempre la característica de esta clase de códigos que es la transición a la mitad del bit ya sea esta de alto a bajo o viceversa.

Una configuración inversa a la indicada es posible, y no habría ninguna desventaja en su uso, en la Figura 3.23 se muestra la representación de bits utilizando el código Manchester Diferencial.

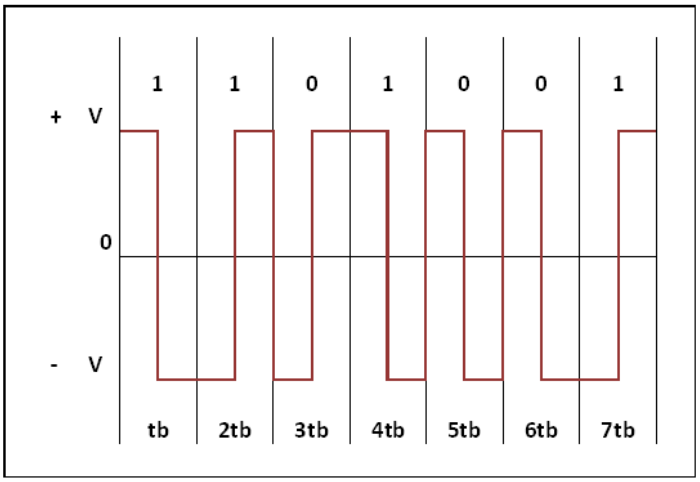


Figura 3.23: Representación de bits utilizando el código Manchester Diferencial

3.3 DISEÑO DEL HARDWARE DEL MODULO

Para el desarrollo del hardware del presente modulo, se utiliza como elemento principal el PIC 18F4550 debido a que este dispositivo permite la conexión a través del puerto USB con el software LabView que necesario en el presente proyecto, el listado de los materiales utilizados se lo puede ver en el Anexo 6.

3.3.1 DISEÑO DEL MODULO CODIFICADOR

Este equipo tiene por objeto realizar la codificación de línea de los datos ingresados por el usuario, el mismo que presenta las siguientes características generales:

- El modulo codificador permite que el usuario ingrese los datos que van a ser codificados a través de una computadora mediante conexión USB o por el teclado que tiene incorporado.
- El modulo tiene una salida de conexión al osciloscopio con el objetivo de poder visualizar la forma en que se están codificando los datos con los diferentes métodos de codificación de línea implementados.

3.3.1.1 Diagrama Esquemático

Para el desarrollo del módulo codificador, se utilizó el dispositivo Multiplexor y Demultiplexor simple de 8 canales el dispositivo CD4053BC cuyo funcionamiento y

características se detallan en el Anexo 3, este elemento se utiliza con el objetivo de manejar los valores de tensión positivos y negativos en el PIC los cuales son necesarios para poder realizar los diferentes tipos de codificación de línea.

El diagrama esquemático del módulo codificador se muestra en la Figura 3.24.

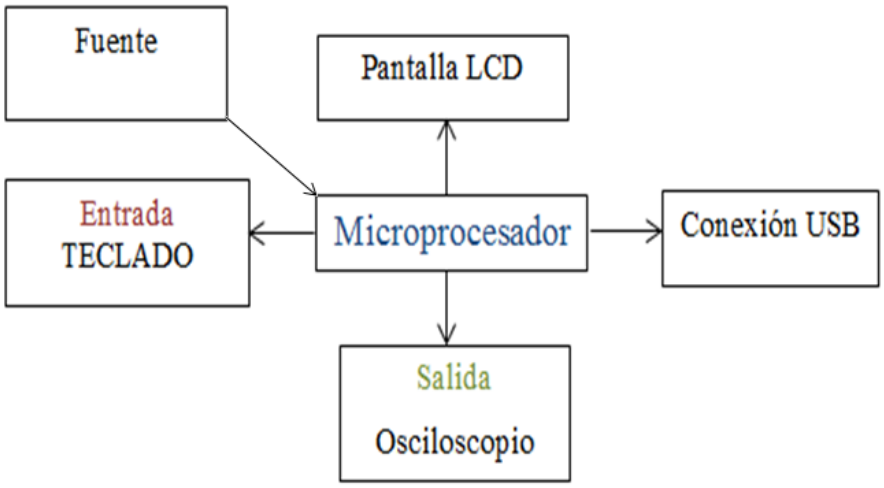
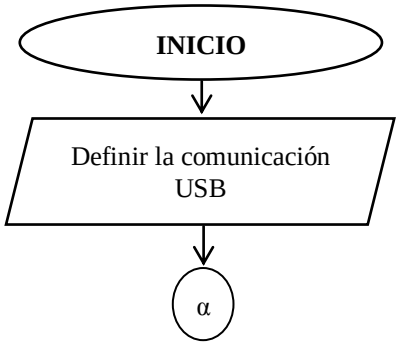


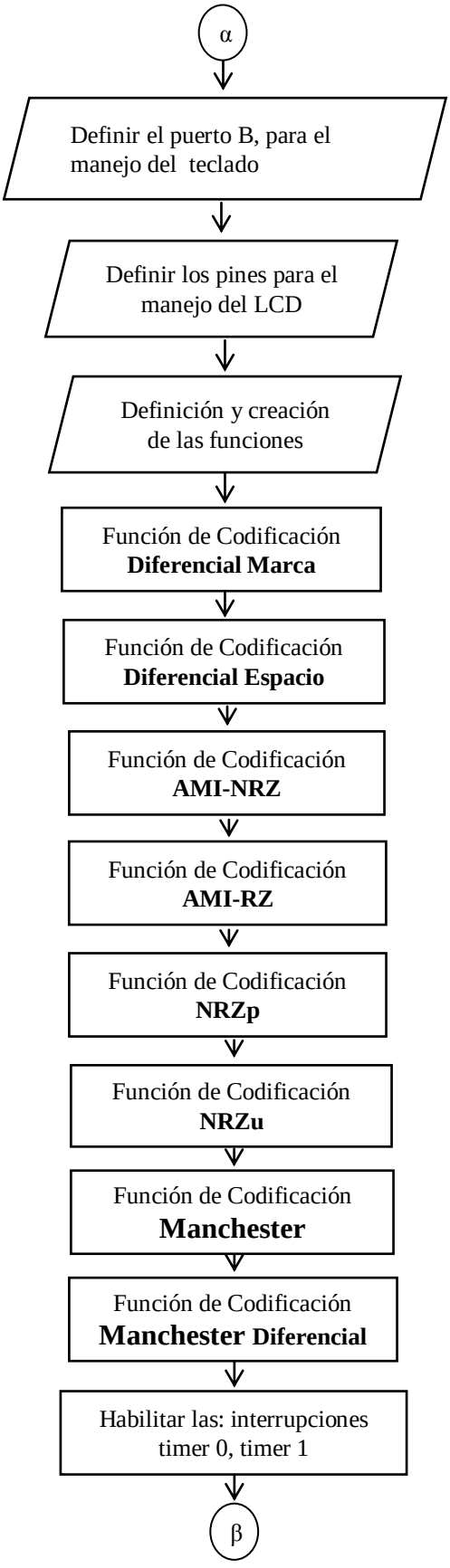
Figura 3.24: Diagrama de Bloques del Módulo codificador

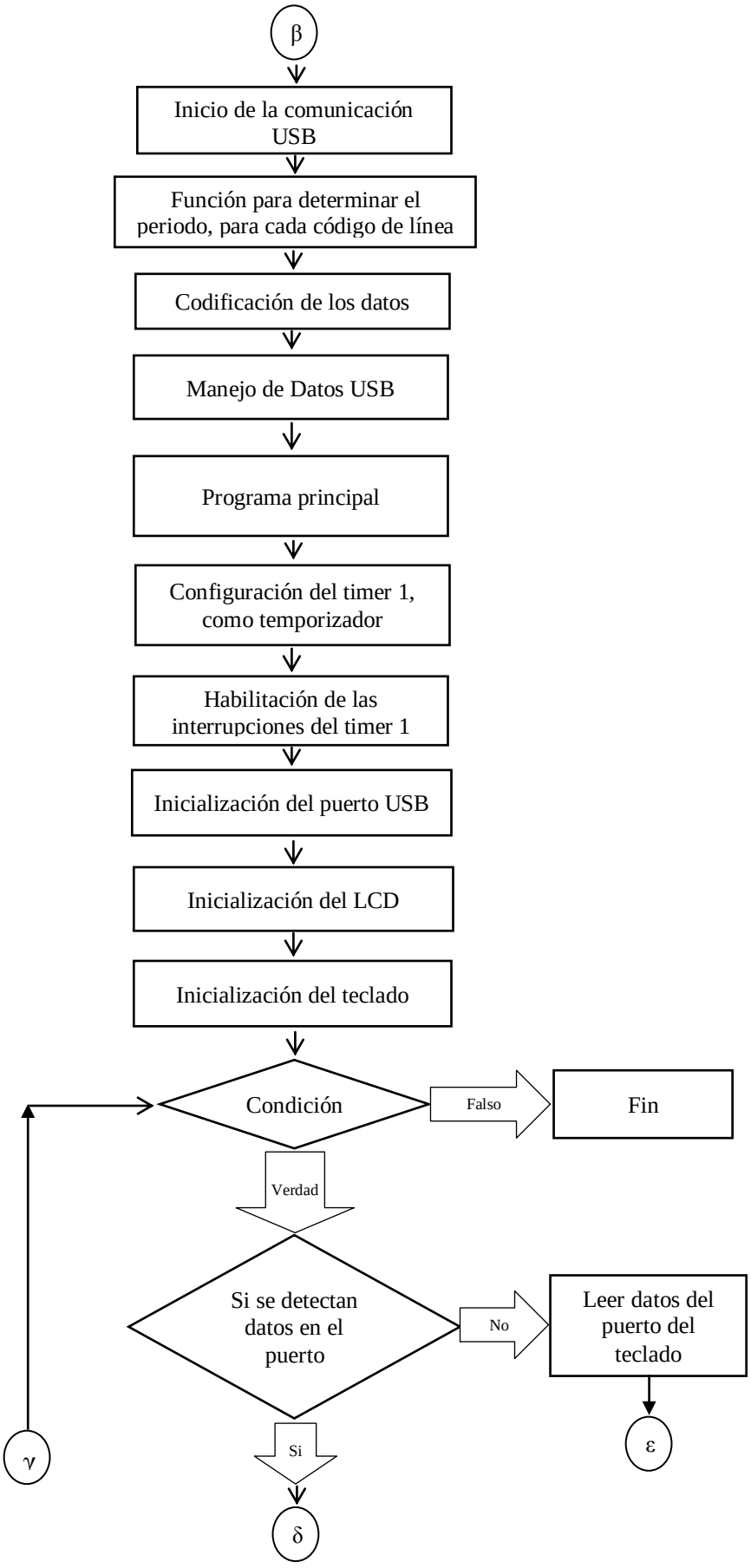
3.3.1.2 Desarrollo del programa para el Microcontrolador

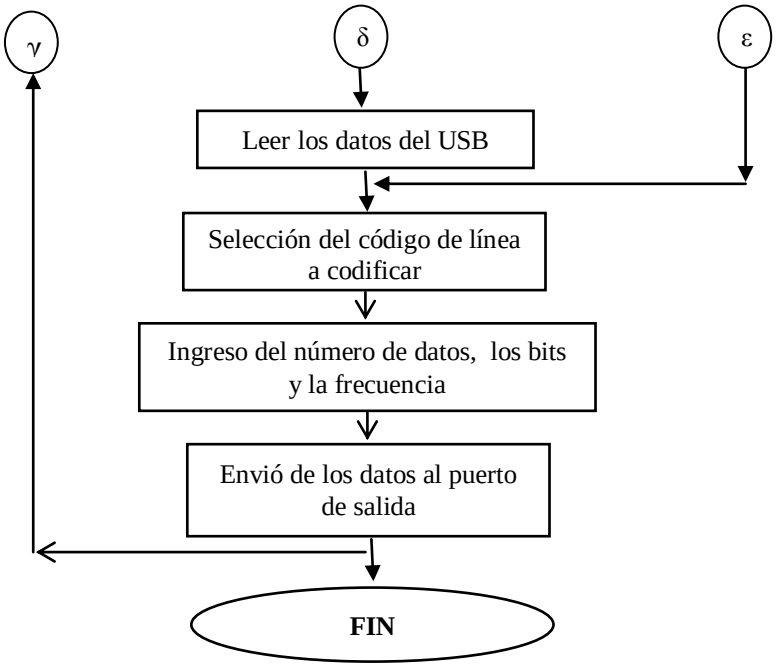
El programa mediante el cual se desarrollará el programa para que el Microcontrolador pueda realizar la función de codificación de línea de los datos es el software PICC debido a que este compilador presenta una ventaja en relación a los demás compiladores una de ellas es la de permitir la creación de drivers, para que se pueda desarrollar la conexión USB entre el PIC y la computadora, para diferentes sistemas operativos, a continuación, se muestra el Diagrama de Flujo del programa.

Diagrama de Flujo




α
Definir el puerto B, para el manejo del teclado
Definir los pines para el manejo del LCD
Definición y creación de las funciones
Función de Codificación **Diferencial Marca**
Función de Codificación **Diferencial Espacio**
Función de Codificación **AMI-NRZ**
Función de Codificación **AMI-RZ**
Función de Codificación **NRZp**
Función de Codificación **NRZu**
Función de Codificación **Manchester**
Función de Codificación **Manchester Diferencial**
Habilitar las: interrupciones timer 0, timer 1
β


β
Inicio de la comunicación USB
Función para determinar el periodo, para cada código de línea
Codificación de los datos
Manejo de Datos USB
Programa principal
Configuración del timer 1, como temporizador
Habilitación de las interrupciones del timer 1
Inicialización del puerto USB
Inicialización del LCD
Inicialización del teclado
Condición
Falso
Fin
Verdad
Si se detectan datos en el puerto
No
Leer datos del puerto del teclado
ε
γ
Si
δ



En el diagrama de flujo anterior, se muestran los diferentes procesos que forman parte del programa del codificador, el programa desarrollado en el software PICC para ser grabado en el Microprocesador PIC 18F4550 se encuentra en el Anexo 4.

3.3.1.3 Simulación

En la Figura 3.25 se muestra la simulación realizada para el módulo codificador con la ayuda del software Proteus mediante el cual se comprobó su funcionamiento antes de su implementación.

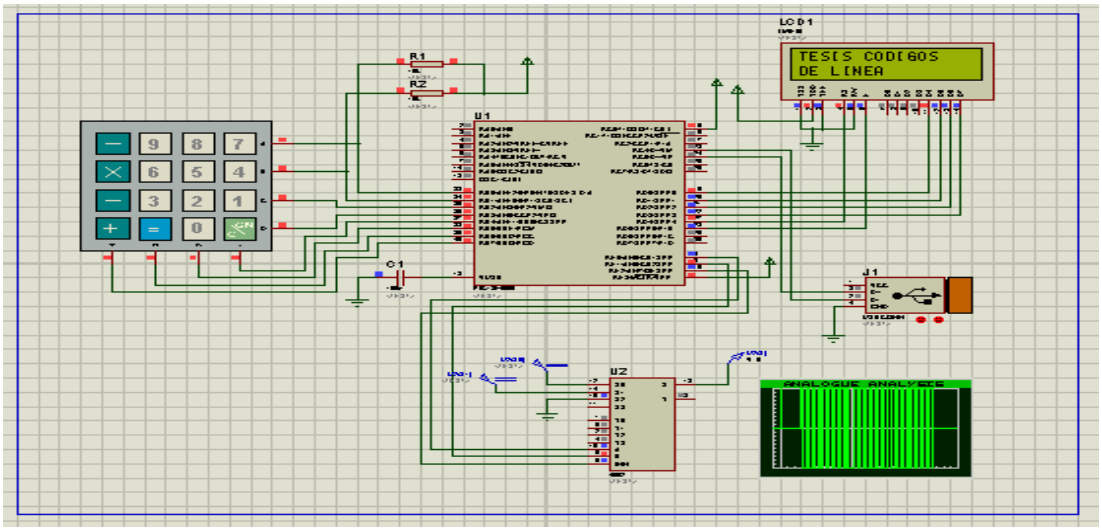


Figura 3.25 Simulación módulo del codificador

3.3.2 DISEÑO DEL MODULO DECODIFICADOR

Este módulo tiene la función de realizar la decodificación de los datos enviados desde el modulo codificador, este equipo presenta las siguientes características:

- El modulo decodificador permite que el usuario seleccione el código de línea de acuerdo al utilizado en el módulo codificador.
- El modulo tiene una conexión mediante el puerto USB para visualizar los datos en el software LabView.
- La decodificación de los datos es presentada al usuario en formato hexadecimal en la pantalla del módulo.

3.3.2.1 Diagrama Esquemático

El diagrama esquemático de las partes que conforman el módulo decodificador desarrollado se lo muestra en la Figura 3.26, el cual como se puede observar tiene una pantalla para visualizar los datos recibidos, permite seleccionar el código con el cual va a trabajar, y los elementos para reiniciar el equipo.

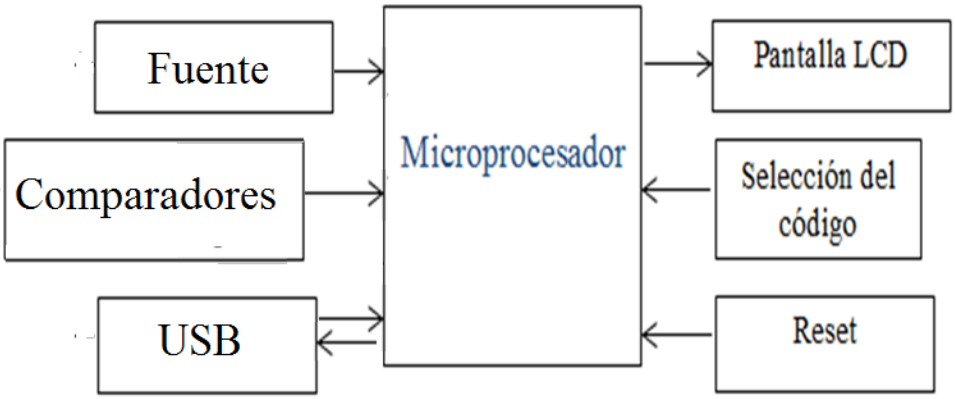


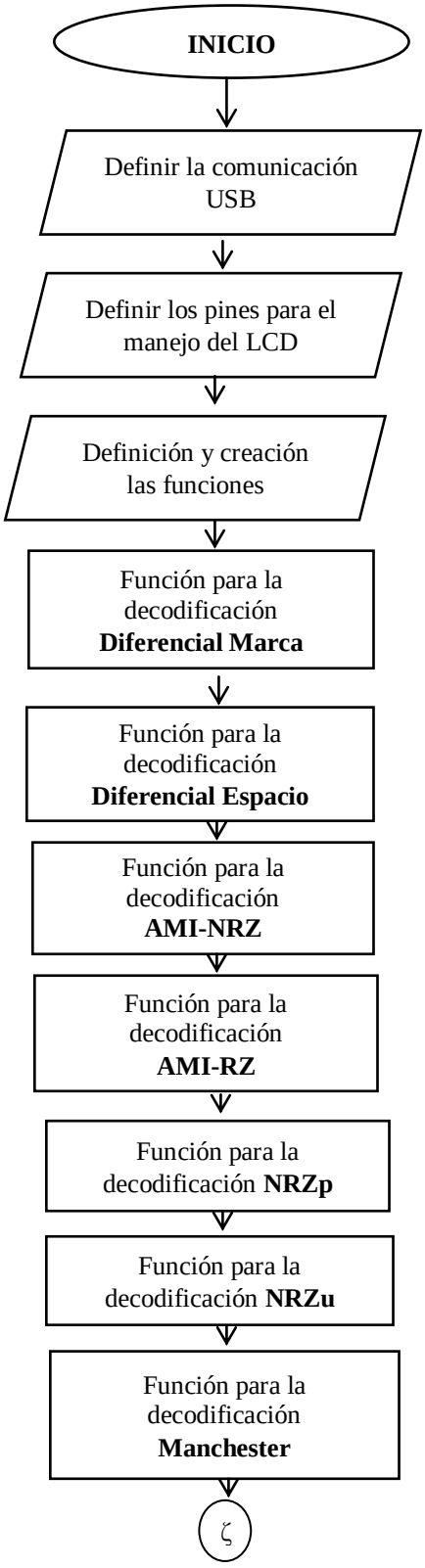
Figura 3.26: Diagrama de Bloques del Módulo Decodificador

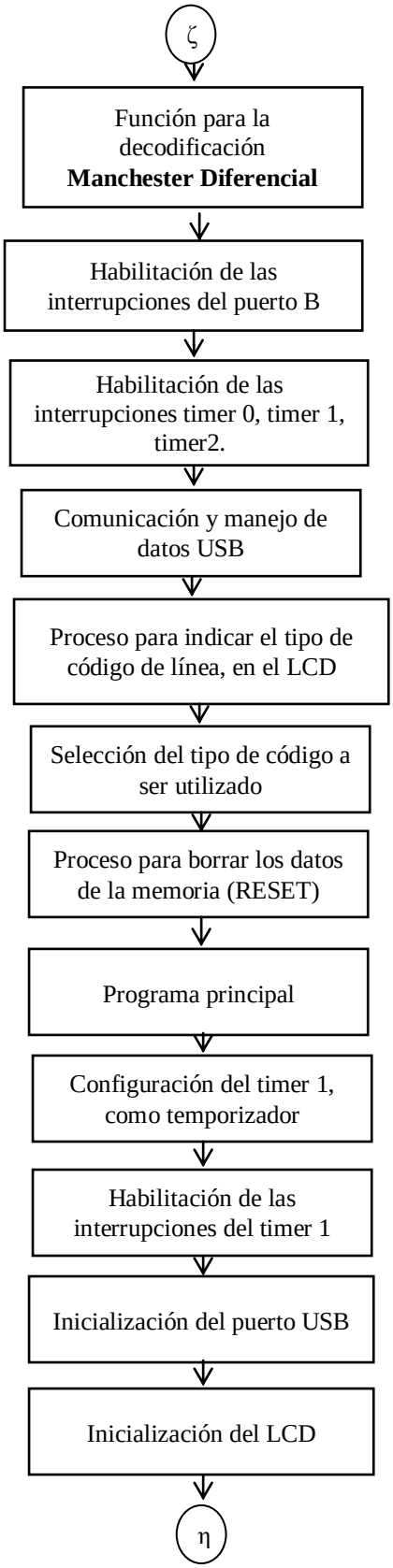
3.3.2.2 Desarrollo del programa para el Microcontrolador

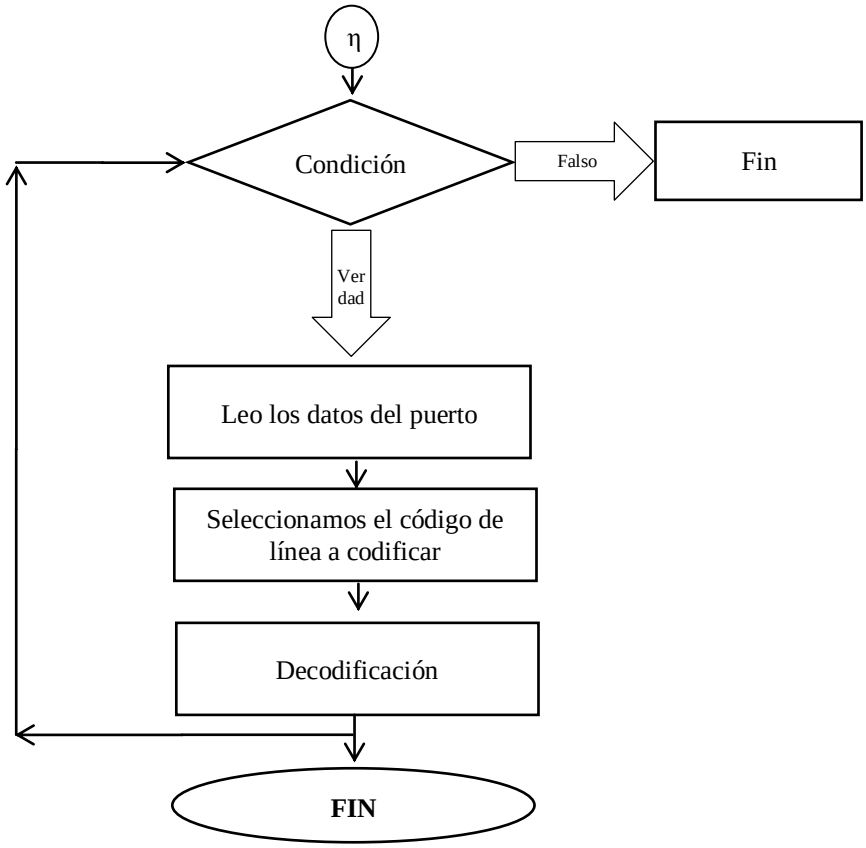
El programa diseñado para el Microcontrolador de este módulo permitirá realizar la decodificación de los datos recibidos desde el módulo codificador que al igual que este último será desarrollado en el software PICC.

El Diagrama de Flujo correspondiente a este módulo se muestra a continuación.

Diagrama de Flujo




ζ
Función para la decodificación **Manchester Diferencial**
Habilitación de las interrupciones del puerto B
Habilitación de las interrupciones timer 0, timer 1, timer2.
Comunicación y manejo de datos USB
Proceso para indicar el tipo de código de línea, en el LCD
Selección del tipo de código a ser utilizado
Proceso para borrar los datos de la memoria (RESET)
Programa principal
Configuración del timer 1, como temporizador
Habilitación de las interrupciones del timer 1
Inicialización del puerto USB
Inicialización del LCD
η



En el diagrama de flujo mostrado anteriormente, se observan los diferentes procesos que forman parte del programa para decodificar los datos, el programa desarrollado en el software PIC C y que va a ser grabado en el microprocesador PIC 18F4550 se encuentra en el Anexo 5.

3.3.2.3 Simulación

En la Figura 3.27 se muestra la simulación realizada del módulo decodificador con la ayuda del software Proteus mediante el cual se comprueba su funcionamiento antes de la implementación del mismo.

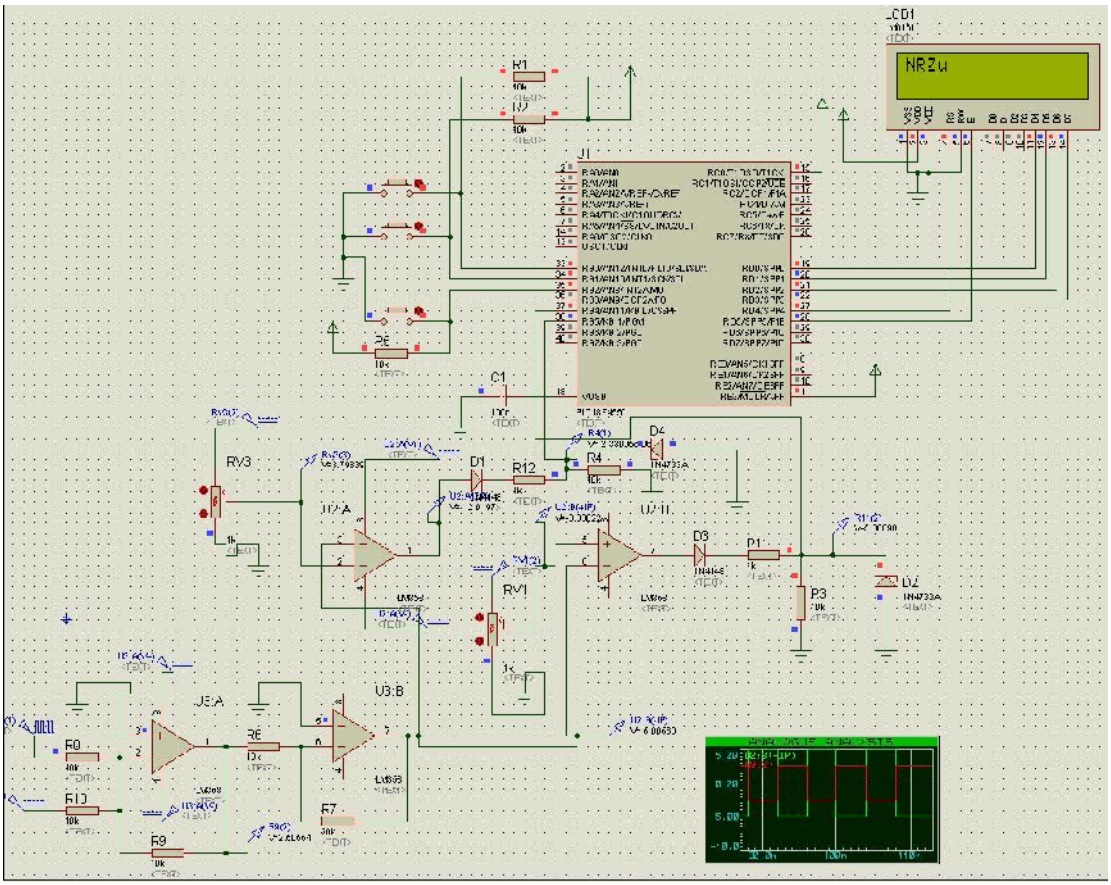


Figura 3. 27: Simulación del módulo decodificador

3.4 ELABORACION DEL SOFTWARE DEL MODULO

En el desarrollo del programa que permitirá desde la computadora el manejo del módulo codificador y decodificador de códigos de línea materia del presente proyecto se utilizará el software de LabView en su versión 8.5, el programa desarrollado utiliza una característica que proporciona este lenguaje de programación gráfica conocida como SubVI de sus siglas sub Virtual Instrument o sub Instrumento Virtual esto permitirá dividir al programa principal en varios sub programas facilitando el desarrollo del mismo, adicionalmente para poder indicar la forma en que se desarrollo el programa se van a utilizar los nombres en ingles de los controles e indicadores del Panel Frontal y de las Funciones del Diagrama de Bloques que se utilicen.

3.4.1 DESARROLLO SUBVI PARA EL CODIGO DE LINEA NRZu

Para la creación de este SubVI se necesita definir los parámetros de entrada necesarios para la codificación estos datos son la frecuencia de muestreo, la frecuencia del bit, y el numero de bits que van a ser codificados los parámetros de la frecuencia de muestreo y de bit son un **Numeric Control** del menú **Controls** del **Front Panel** y al número de bits como un **Array** de un **Numeric Control** del mismo menú cuyo tamaño varía de acuerdo al número de datos que se ingresen, estos controles van a tener su equivalente en el **Block Diagram** como se muestra en la Figura 3.28.

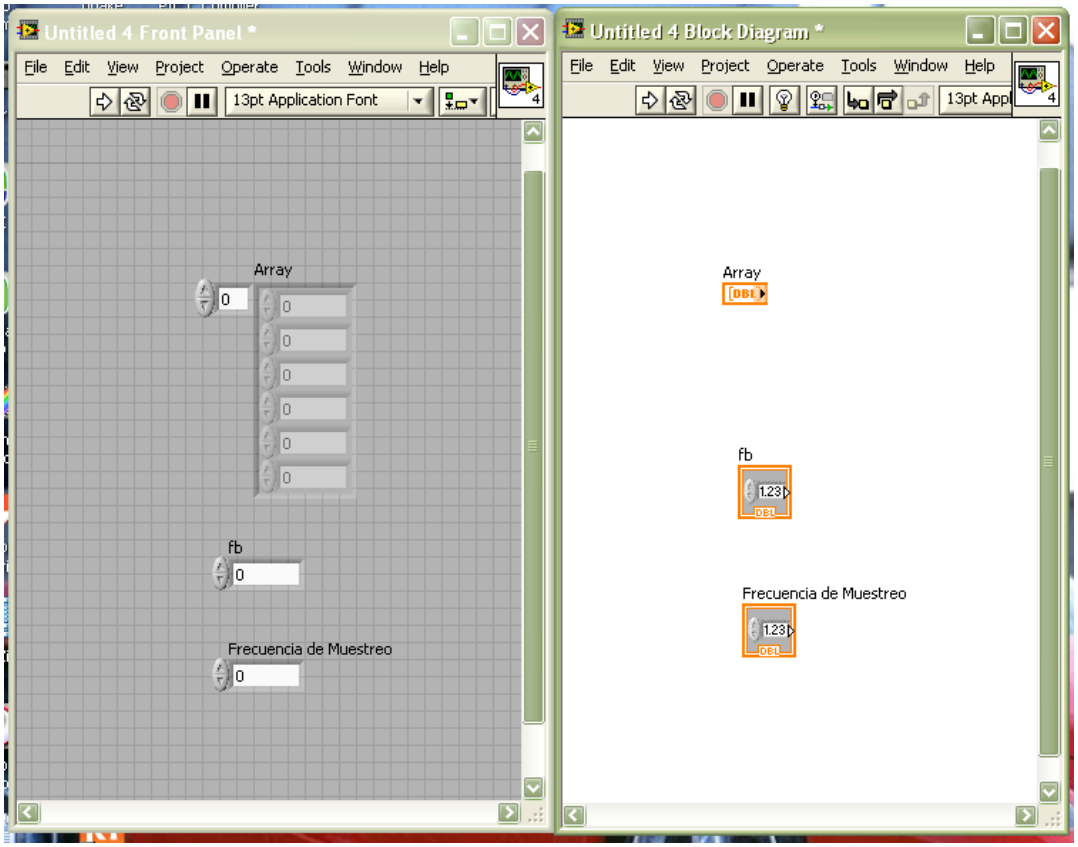


Figura 3.28 Definición de los parámetros de entrada código NRZu.

Una vez que ya se han definido los controles de los parámetros en el **Front Panel**, en el **Block Diagram** se desarrollan las diferentes funciones que van a permitir realizar la codificación NRZu siguiendo el algoritmo de codificación de este código.

Primero se va a obtener una grafica que represente la Señal de entrada en bits, esta gráfica indicará la señal aleatoria en bits que entra al codificador para lo cual se modifica el tipo de representación del **Array** de **Double Presicion** a **Unsigned Byte** y se conecta a un **Array Size** para poder ingresar a un **For Loop** en donde se convierte el numero de datos en número de bits esta parte del programa se muestra en la Figura 3.29, en donde el **Numeric Array** se cambia por un Arreglo Booleano mediante la función **Numeric to Boolean Array** y este se conecta hacia un Conversor de Booleano a 0 y 1 con la función **Boolean to (0,1)** para almacenar cada bit en un arreglo mediante la función **Insert into Array** en donde se guardan los datos de acuerdo a un índice y cada arreglo tendrá un tamaño de 8 que equivale a 1 byte.

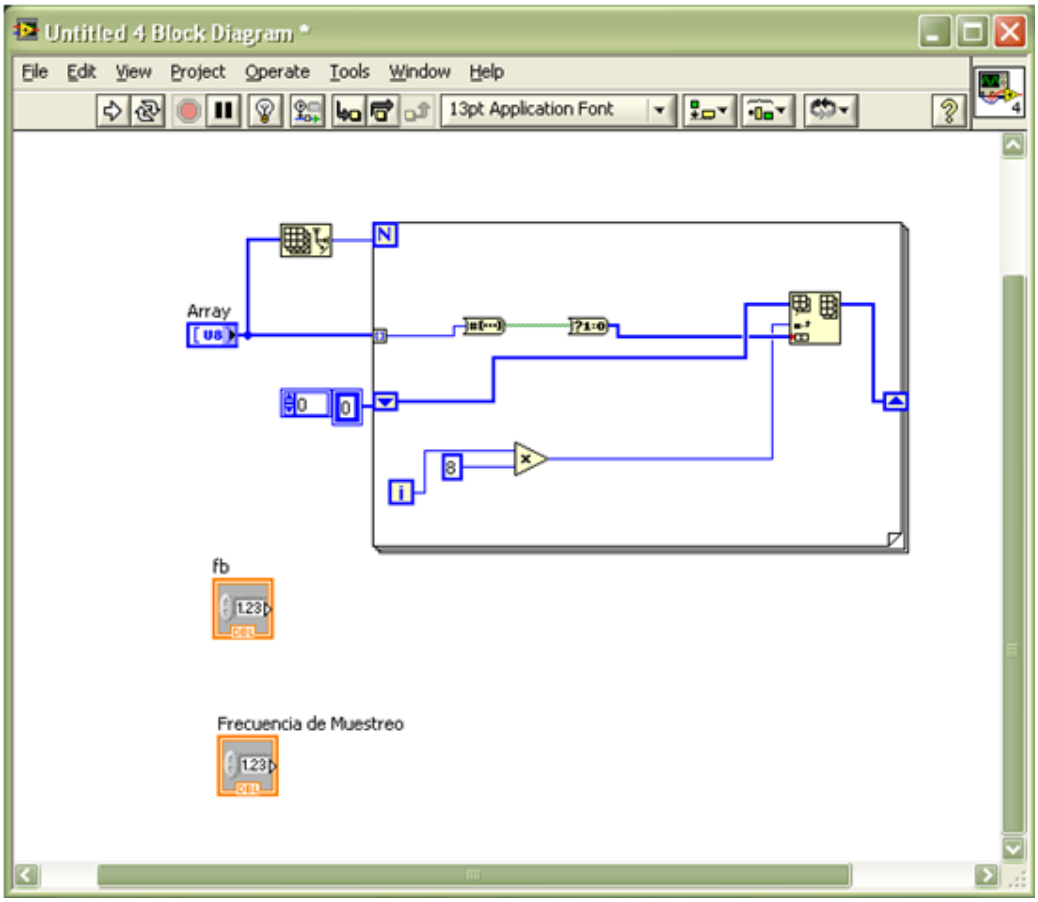


Figura 3.29: Conversión de los datos de entrada.

Ahora la salida del **Insert Into Array** se conecta primero a un **Array Size** el cual devolverá un valor que indicará el número de elementos que se deben a graficar y servirá para la cuenta del nuevo **For Loop**, por ultimo esta salida se conecta a un

Index Array dentro de este **For Loop** donde se relaciona la frecuencia de muestreo, la frecuencia de bit y el número de elementos, estos datos son ingresados en un **Case Structure** para cumplir con el algoritmo de asignar un valor de +V o -V al símbolo "1" mientras que al símbolo "0" se le asignara el valor de 0V, por último la salida del **Initialize Array** que se encuentra dentro del **Case Structure** se conecta a un **Reshape Array** para dimensionar una nueva matriz de acuerdo a los valores que se obtuvieron dentro de este **For Loop** como se muestra en la Figura 3.30.

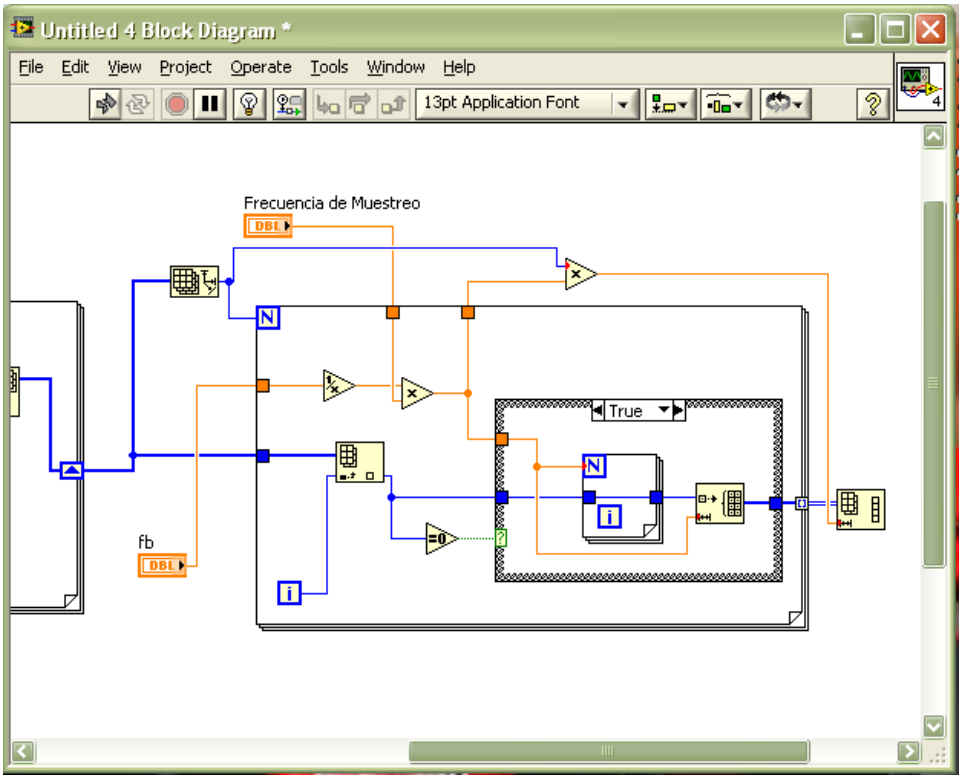


Figura 3.30: Codificación NRZu

Para obtener la onda cuadrada que representara la señal de entrada en bits en base a los parámetros antes definidos, se lo hará con la función **Square Waveform** en donde ingresan los datos de la frecuencia de bit, la frecuencia de muestreo y el dato del **Reshape Array** obtenido en el parte anterior, la salida del **Square Waveform** se conecta a la función **Get Waveform Components** esta función se encarga de dar la forma de la onda en base a las características antes definidas, para poder construir la onda con los parámetros modificados se utiliza la función **Build Waveform** y su salida se conecta al control **Waveform Graph** que lleva como título "Señal de

entrada en bits" permitiendo visualizar esta gráfica en el **Front Panel** tal como se puede observar en la Figura 3.31.

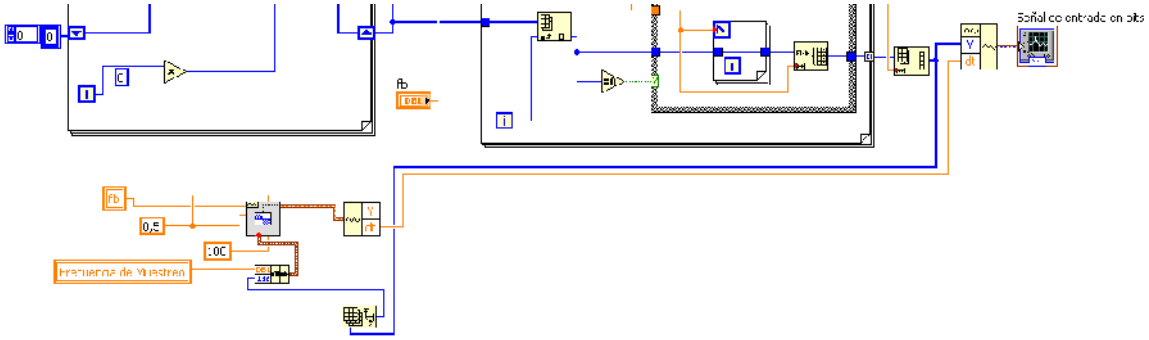


Figura 3.31: Visualización Señal de entrada en bits

Para visualizar la señal codificada en función del tiempo se utilizan los parámetros anteriormente obtenidos y con la ayuda de la función **Build Waveform** se ingresan los parámetros, esta salida se conecta al Indicador **Waveform Graph** que lleva como título "Señal codificada en tiempo" para obtener la visualización de la Señal Codificada en Tiempo del código NRZu como se muestra en la Figura 3.32.

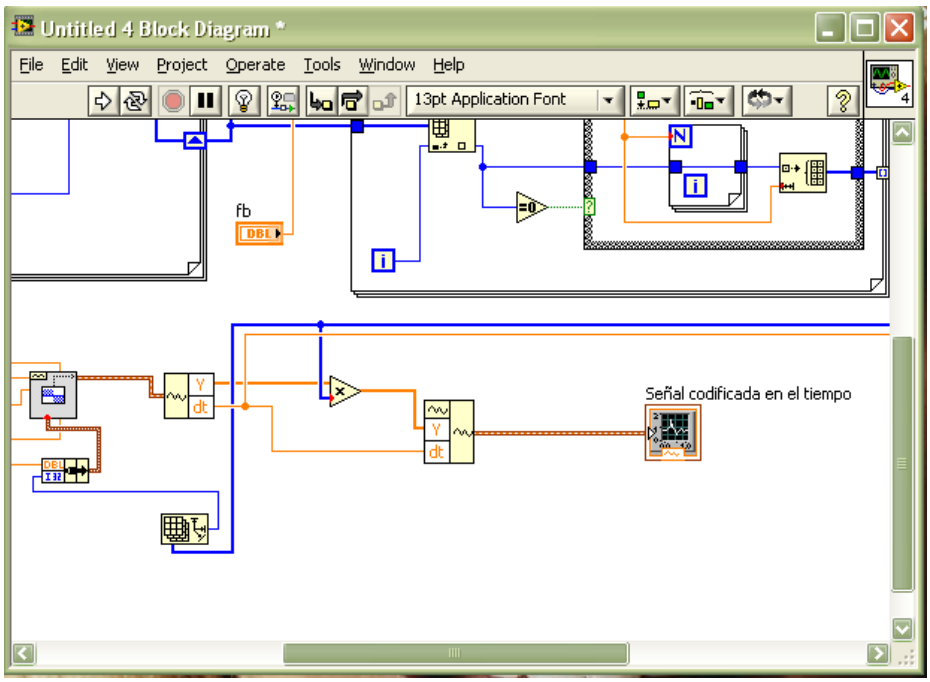


Figura 3.32: Visualización Señal codificada en el tiempo

Para visualizar la señal codificada en el dominio de la frecuencia se usa la función **Spectral Measurement,** se la configura para que grafique la fase del espectro FFT y

muestre los resultados en un Indicador **Waveform Graph** que lleva como título "FFT" como se puede ver en la Figura 3.33.

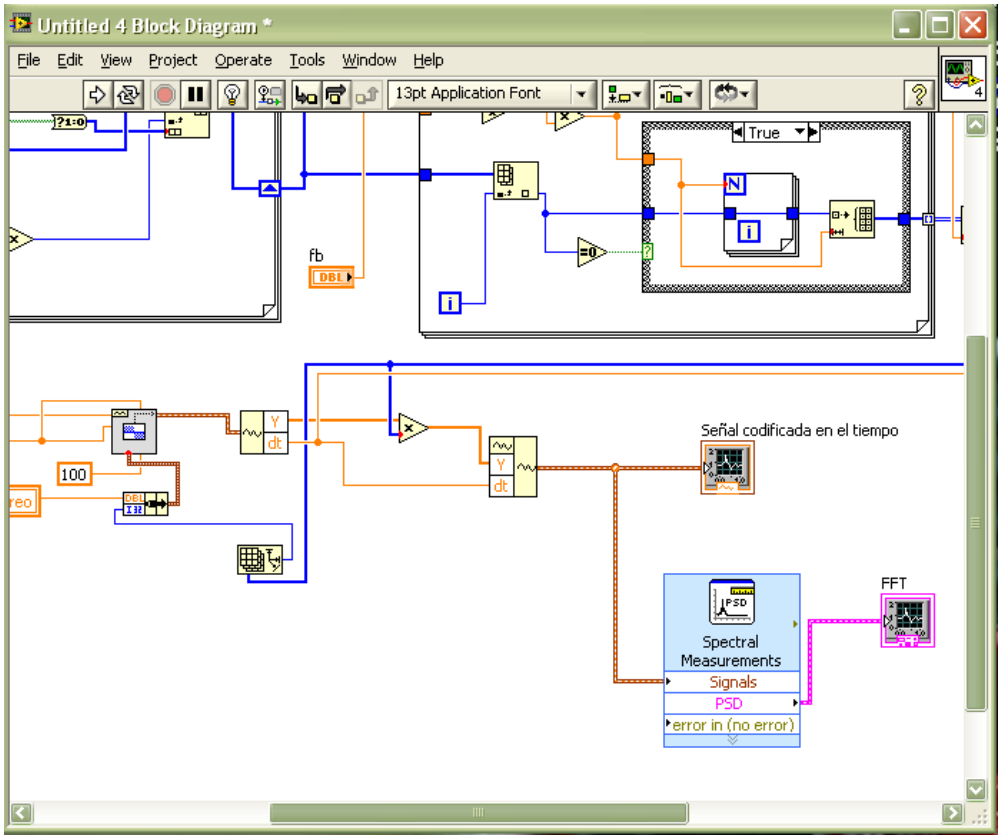


Figura 3.33: Visualización de la señal codificada en el dominio de la frecuencia.

Adicionalmente en este SubVI se desarrollo la programación para visualizar el Diagrama de constelación cuyo diagrama de bloques se muestra en la Figura 3.34.

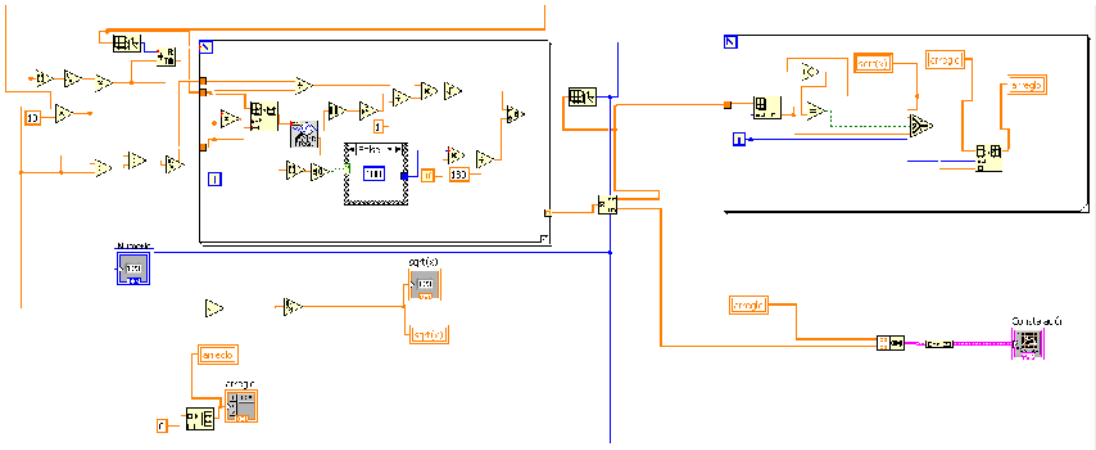


Figura 3.34: Diagrama de Constelación

Se procede a grabar el SubVI con el nombre subvi-NRZu quedando el **Front Panel** y el **Block Diagram** del SubVI para el código de línea NRZu como se muestra en la Figura 3.35 y la Figura 3.36 respectivamente.

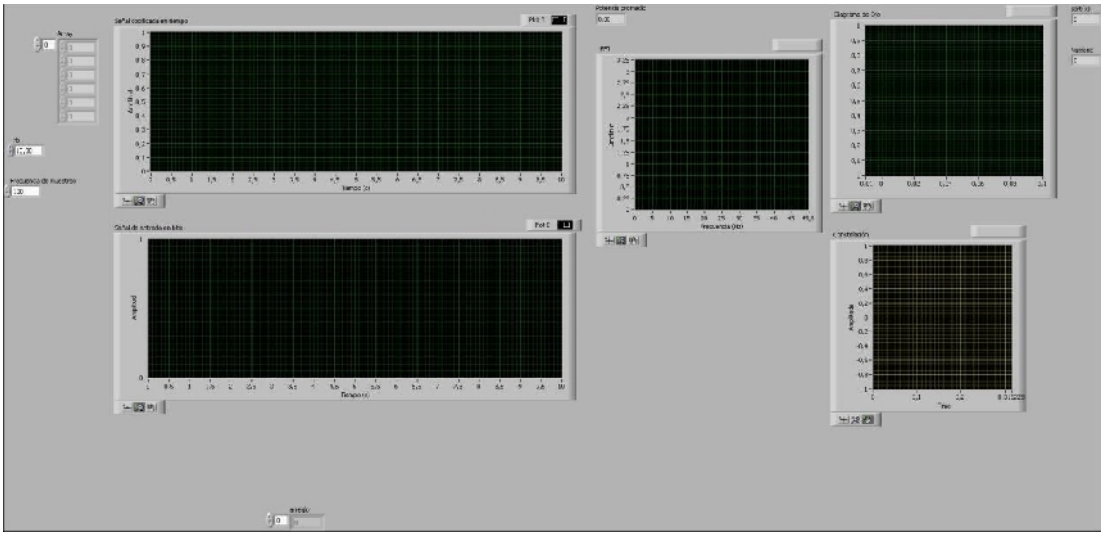


Figura 3.35: Panel Frontal SubVI del Código NRZu

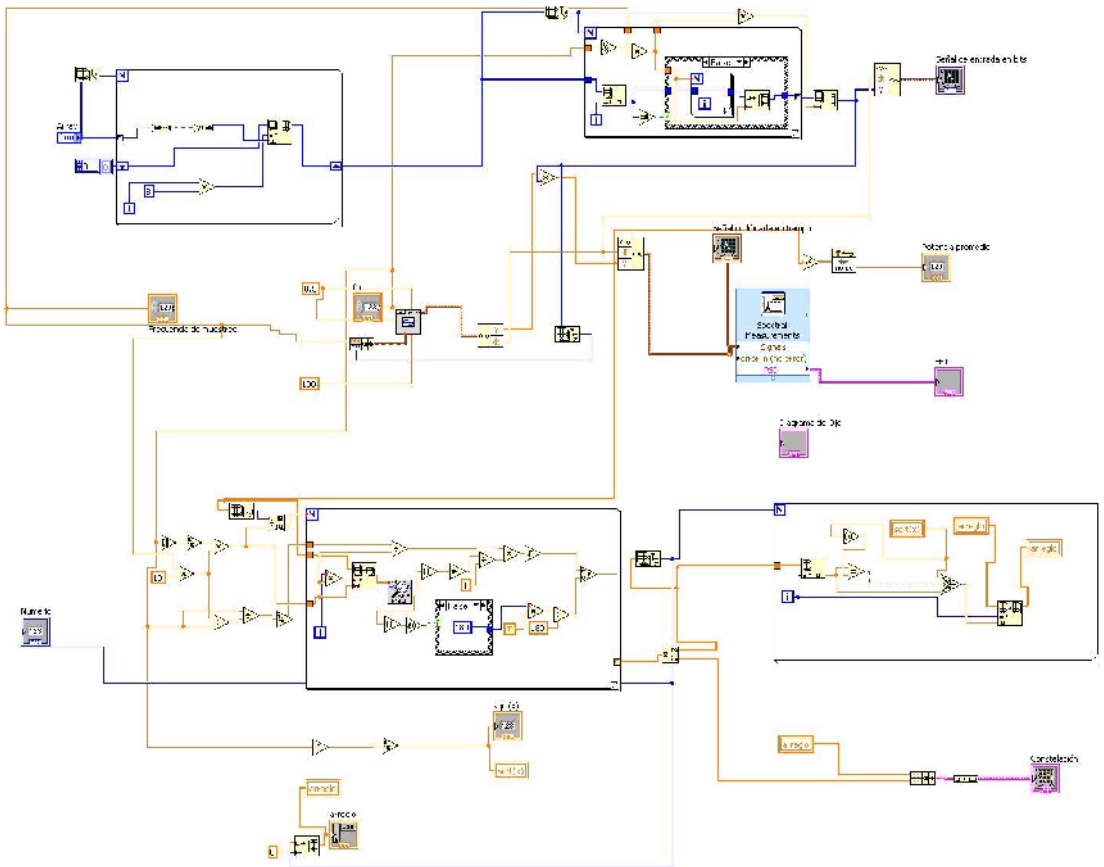


Figura 3.36: Diagrama de Bloques SubVI del Código NRZu.

3.4.2 DESARROLLO SUBVI PARA EL CODIGO DE LINEA NRZp

Para el desarrollo de este SubVI es similar al del SubVI del código NRZu debido a que el algoritmo de codificación es muy similar, se diferencia únicamente en que al símbolo 0 se le asignará el valor de –V, el procedimiento de desarrollo del programa seria:

Se definen los parámetros de entrada como se indica en la Figura 3.37.

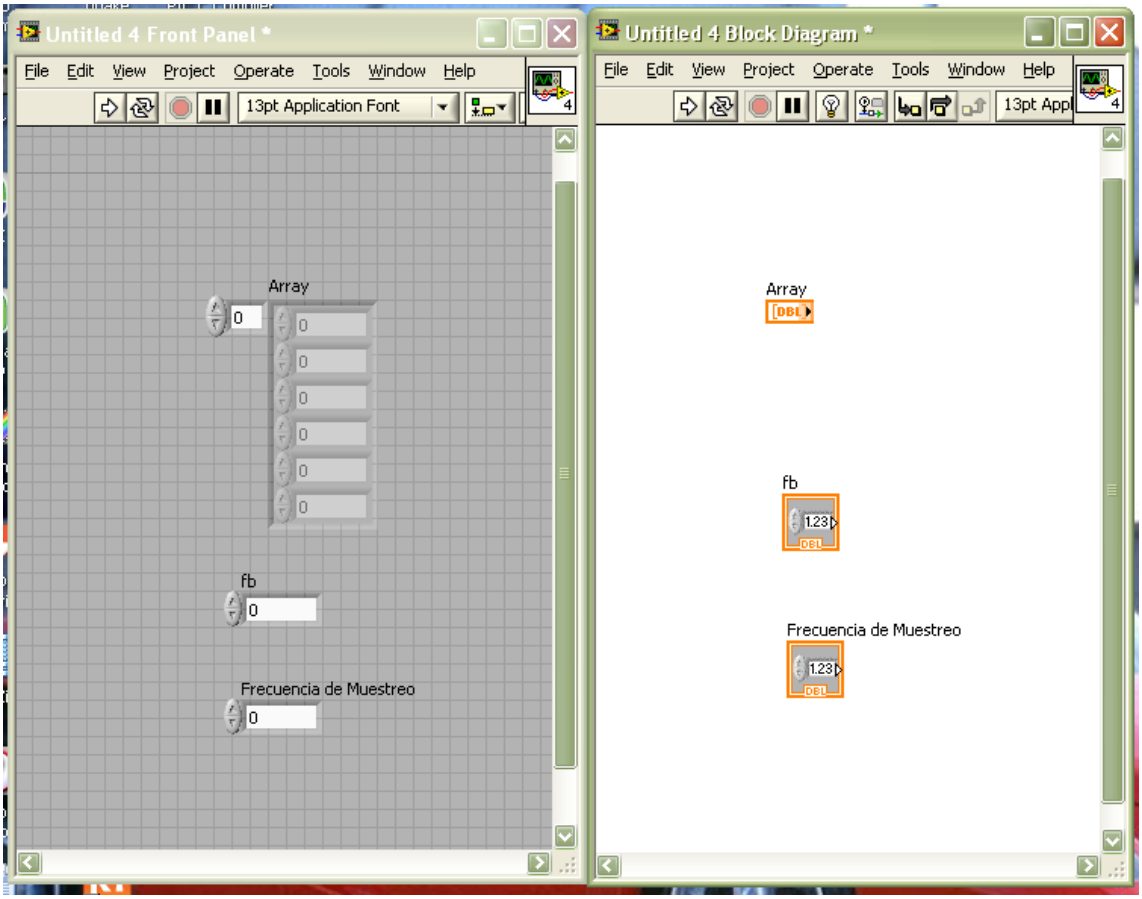


Figura 3.37: Definición de los parámetros de entrada código NRZp.

Se realiza la conversión del arreglo numérico de entrada y las respectivas conversiones en el primer **For Loop**, como se puede observar en la Figura 3.38.

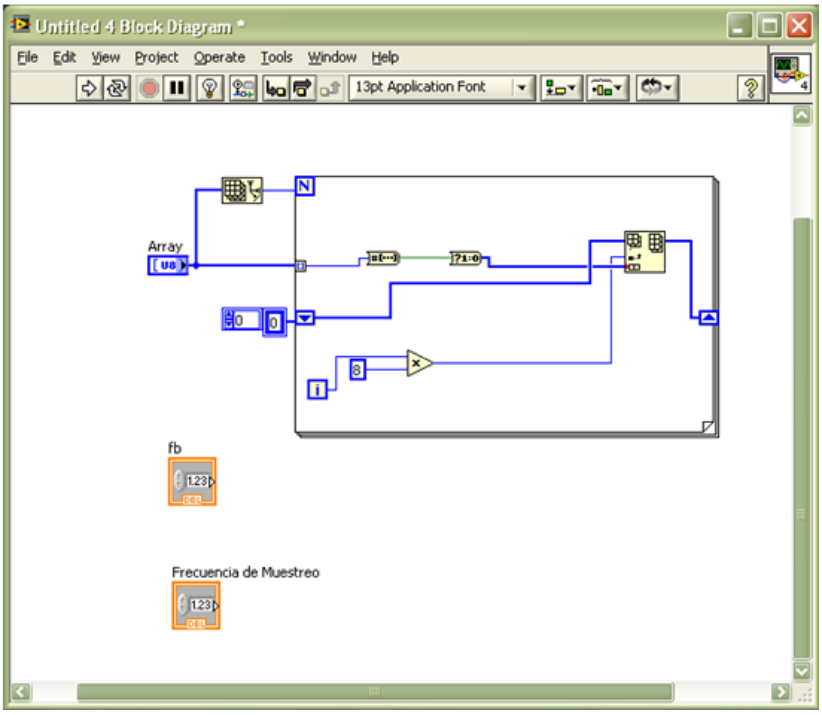


Figura 3.38: Conversión de los datos de entrada

Se sigue los mismos pasos del algoritmo del SubVI del código NRZu para obtener el código NRZp del segundo **For Loop** que realiza la codificación para identificar el número de 1s y 0s y relacionarlos con la frecuencia de bit y de muestreo como se indica en la Figura 3.39.

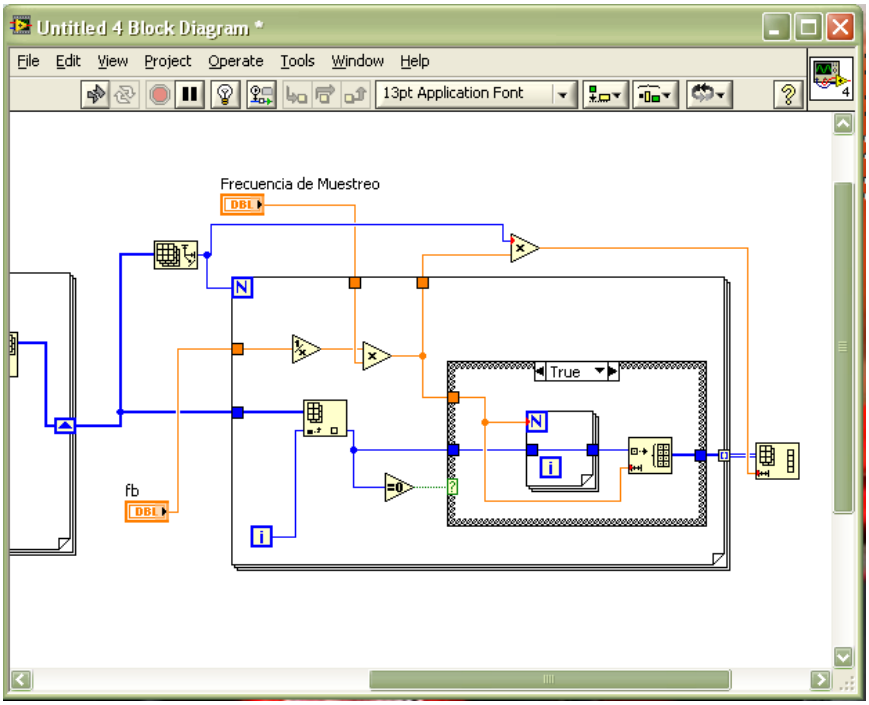


Figura 3.39: Codificación NRZp

Para obtener la visualización de la “Señal de entrada en bits”, se utilizan las mismas funciones que en el SubVI del código NRZu como se puede observar en la Figura 3.40.

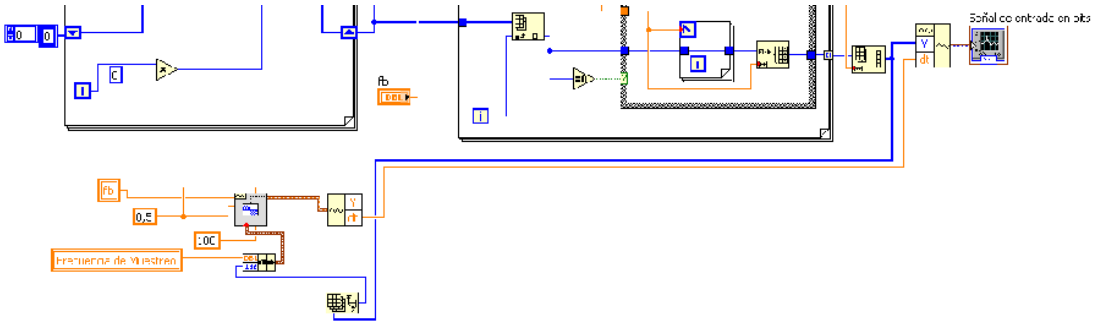


Figura 3.40: Visualización Señal de entrada en bits

Para obtener la gráfica de la señal codificada, a los parámetros que se obtuvieron en el paso anterior se cambia el valor de cero a un valor negativo lo cual se consigue con la función **Negate**, adicionalmente el resultado obtenido del segundo **For Loop** creado se conecta a un **Case Structure** dentro de tercer **For Loop,** en el cual se definen dos contantes la primera es 1 en el case 0 y la segunda es 0 en el case 1 de la función, de esta forma se obtiene la referencia de +V y de –V necesarias para graficar la señal codificada en función del tiempo como se muestra en la Figura 3.41.

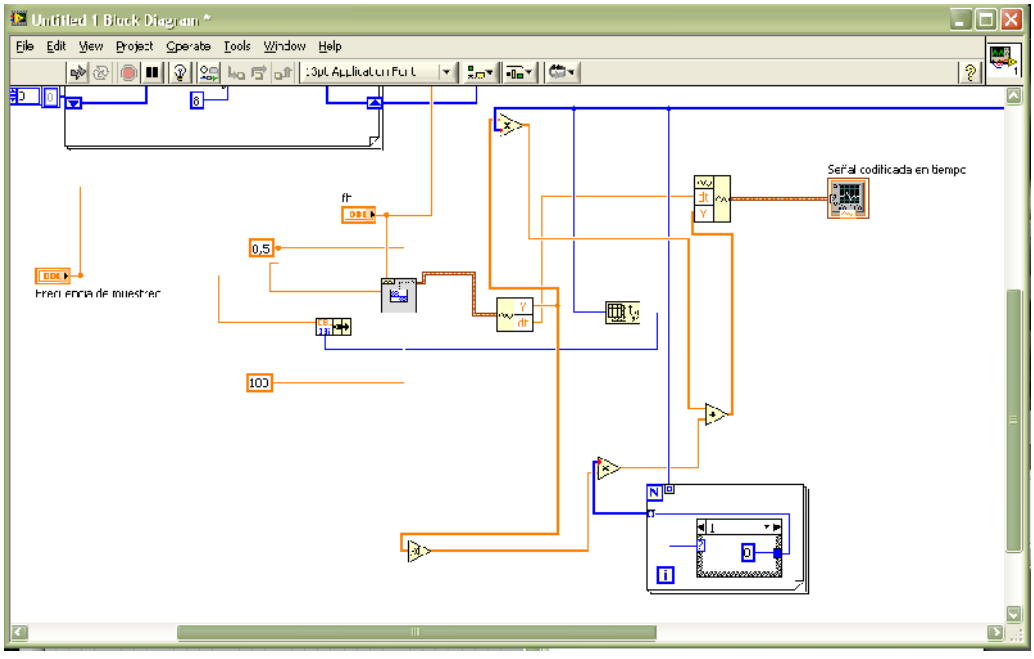


Figura 3.41: Señal codificada en el tiempo en función del tiempo.

Para visualizar la señal codificada en función de la frecuencia se sigue el mismo procedimiento que para el SubVI del código NRZu como se puede verificar en la Figura 3.42.

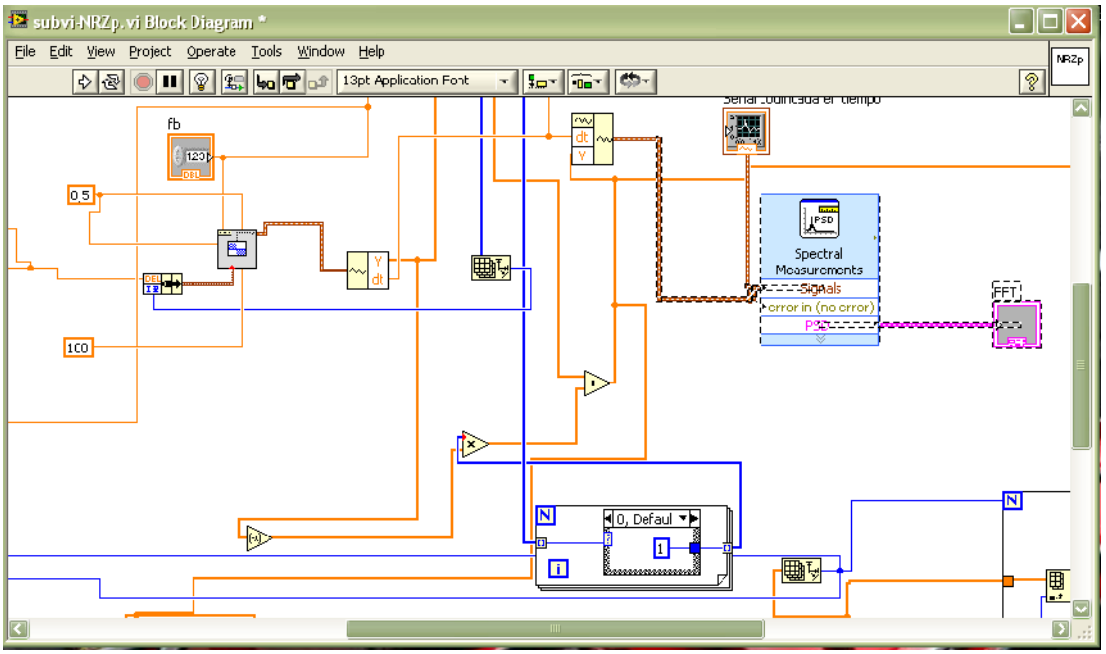


Figura 3.42: Visualización de la Señal codificada en función de la frecuencia.

De la misma forma que para el caso anterior en este SubVI se ha desarrollo la programación para poder visualizar el Diagrama de constelación cuyo diagrama de bloques se muestra en la Figura 3.43.

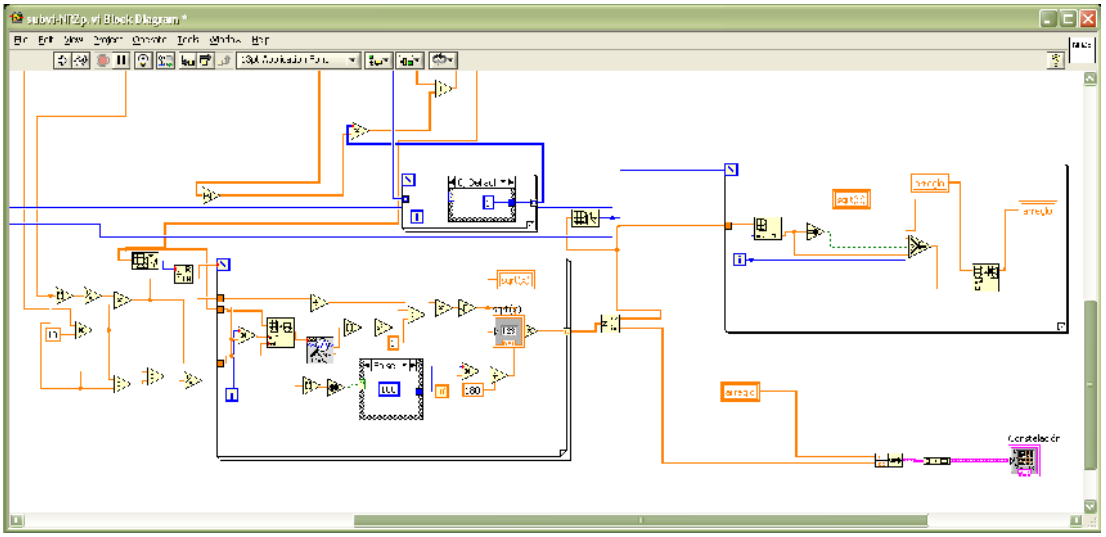


Figura 3.43: Visualización del Diagrama de Constelación.

Por último se procede a guardar el SubVI con el nombre subvi-NRZp quedando el **Front Panel** y el **Block Diagram** del SubVI para el código de línea NRZu como se muestra en la Figura 3.44 y la Figura 3.45 respectivamente.

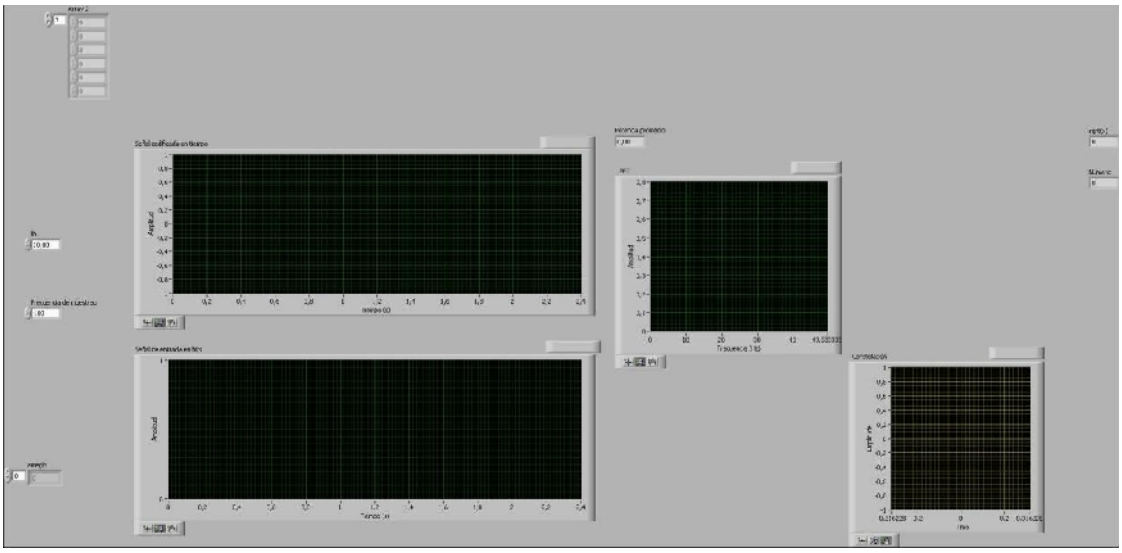


Figura 3.44: Panel Frontal SubVI del Código NRZp

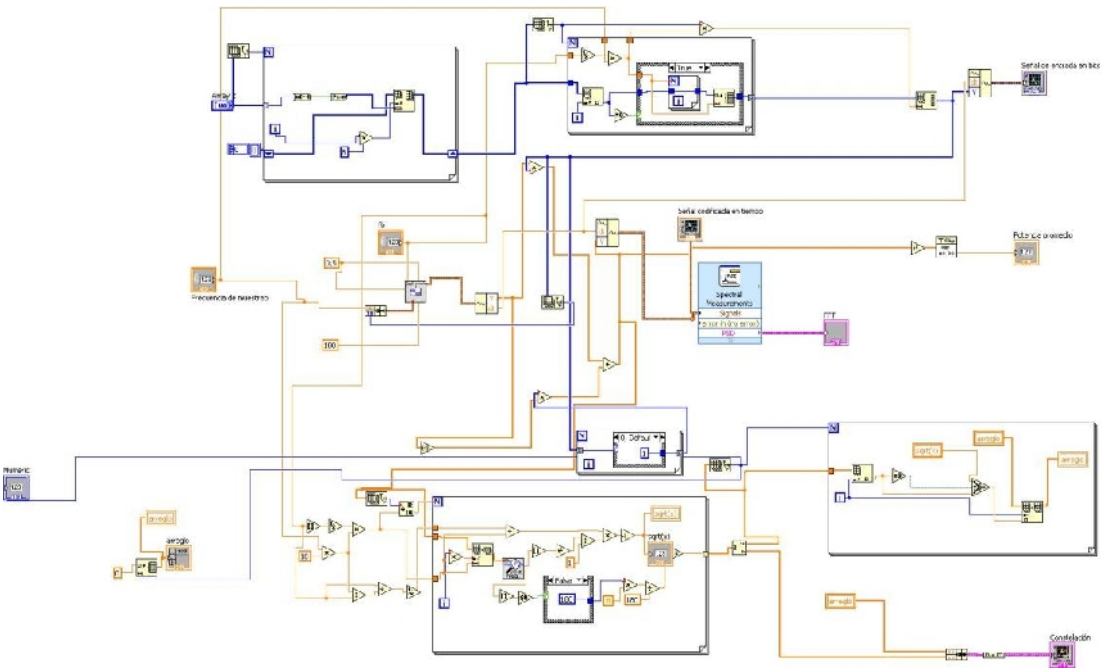


Figura 3.45: Diagrama de Bloques SubVI del Código NRZp.

3.4.3 DESARROLLO SUBVI PARA EL CODIGO DE LINEA DIFERENCIAL ESPACIO

Al igual que para los códigos anteriores se definen los parámetros de entrada tanto en el Front Panel como en el **Block Diagram** como se indica en la Figura 3.46.

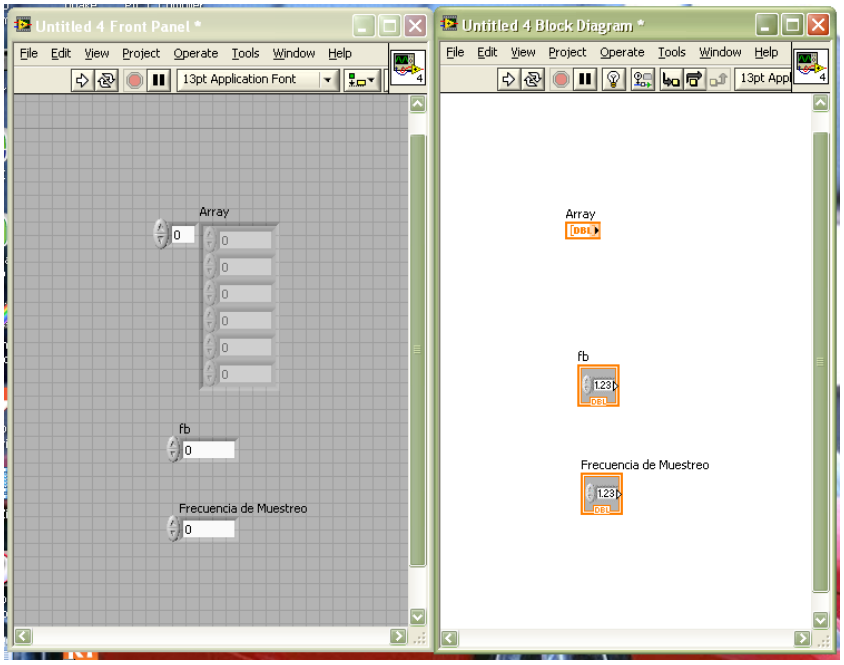


Figura 3.46: Definición de los parámetros de entrada código Diferencial tipo Espacio

Del arreglo inicial se obtiene el número de bits que van a ser codificados, estos datos son almacenados en un segundo arreglo que lleva el título de "Array 2" como se muestran en la Figura 3.47.

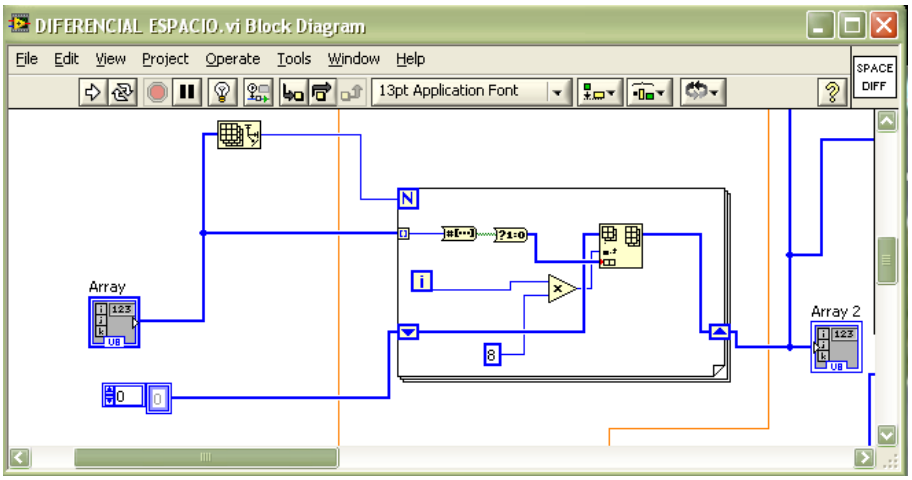


Figura 3.47: Conversión de los datos de entrada

El algoritmo de codificación indica que cuando existe el símbolo 0 se da una transición en los valores de tensión de positivo a negativo o viceversa dependiendo del estado anterior y cuando exista el símbolo 1 no existe transición esta parte de la codificación se la muestra en la Figura 3.48, estos valores son almacenados en un tercer arreglo denominado “Array 3”.

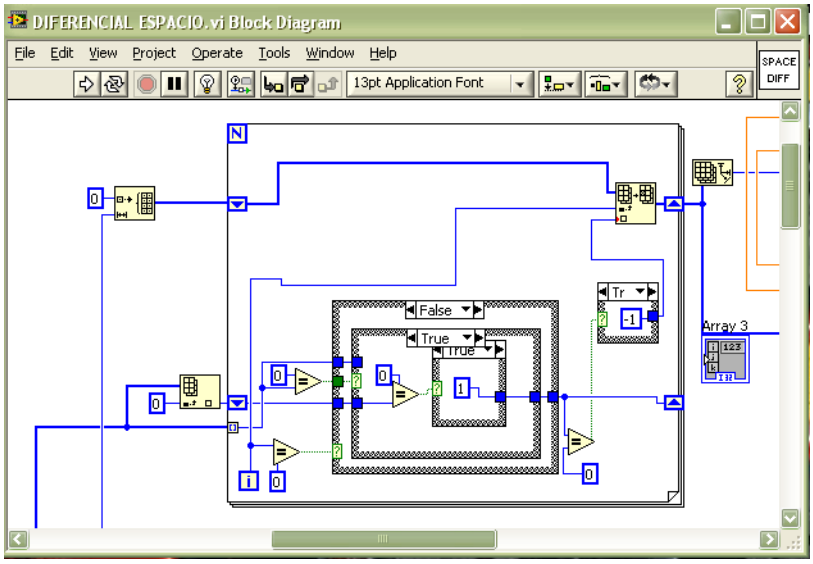


Figura 3.48: Codificación Diferencial Espacio

La relación entre el número de bits, la frecuencia de bit y la frecuencia de muestreo con el objetivo de poder obtener las transiciones de acuerdo al algoritmo del código de línea, se muestra en la Figura 3.49.

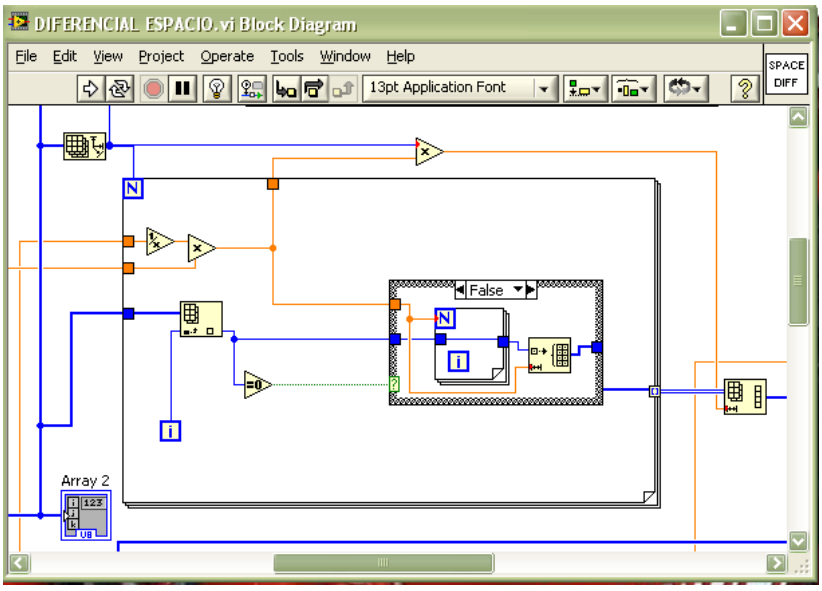


Figura 3.49: Obtención número de transiciones para la Codificación Diferencial Espacio

Para visualizar en el Indicador Titulado “Señal de entrada en bits” es necesario indicar los parámetros que va a tener nuestra nueva onda en base a la frecuencia de muestreo la de bit y el número de bits la parte del programa que realiza esta función se la muestra en la Figura 3.50.

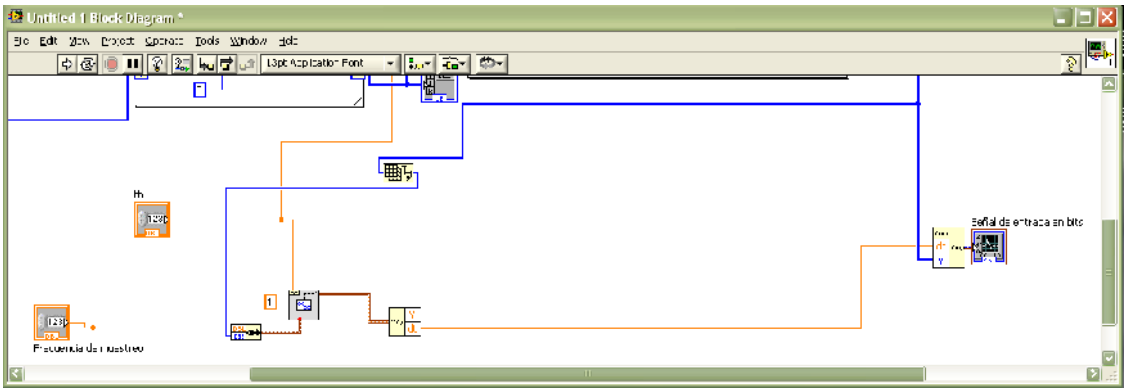


Figura 3.50: Visualización de la Señal de entada en bits.

Para visualizar en el Indicador Titulado “Señal de codificada en tiempo utilizamos los parámetros de la anterior grafica y con la ayuda de la función **Build Waveform** se obtiene la gráfica, la sección del programa que realiza esta función se muestra en la Figura 3.51.

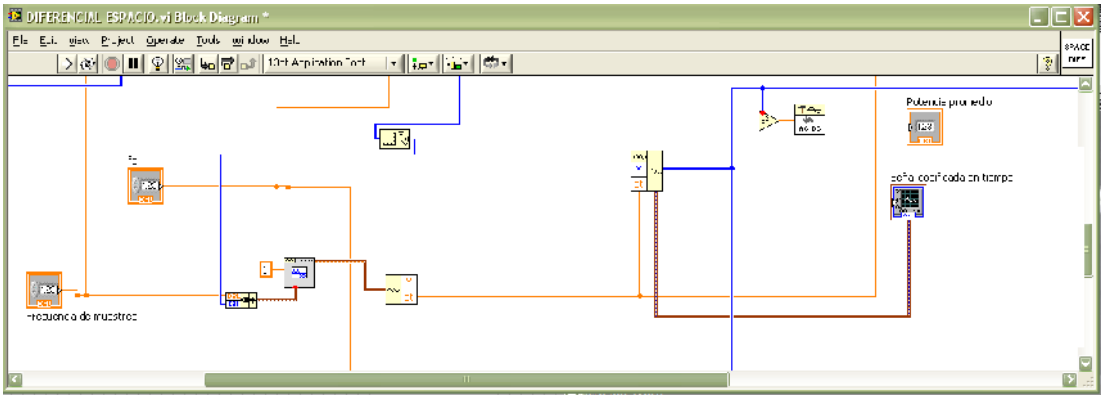


Figura 3.51: Visualización de la señal codificada en función del tiempo.

Ahora con la ayuda de la función **Spectral Measumerments** se obtiene la señal codificada en función de la frecuencia, como se indica en la Figura 3.52.

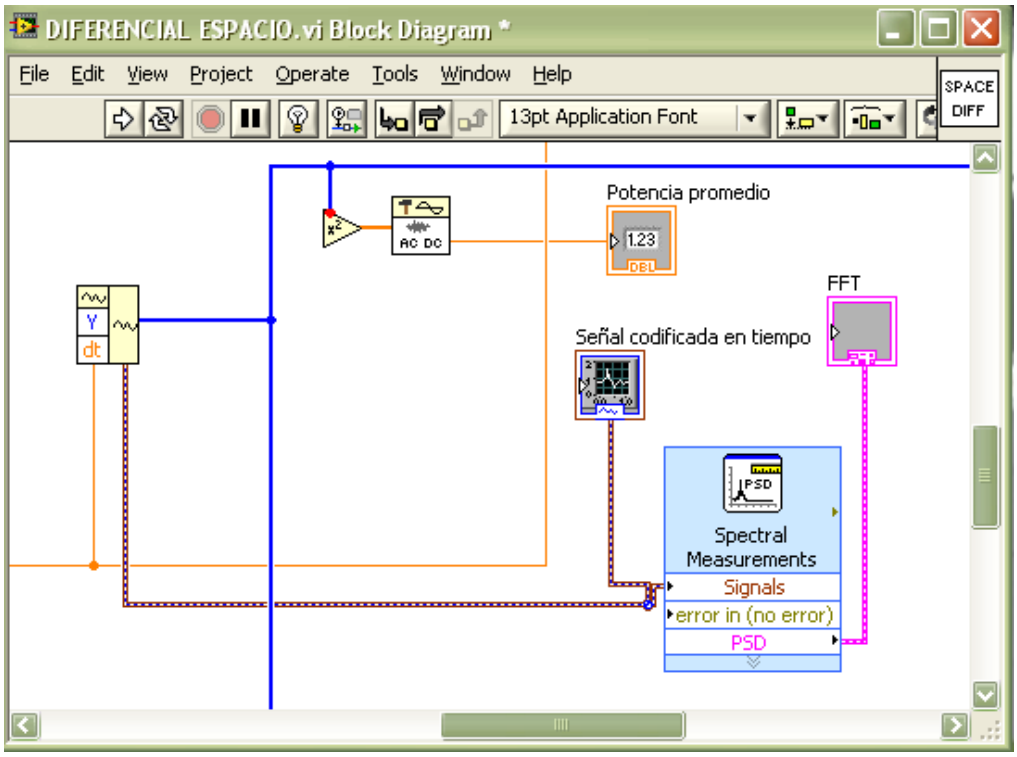


Figura 3.52: Visualización de la señal codificada en función de la frecuencia.

De la misma forma que en los casos anteriores en este SubVI se ha desarrollado visualización del Diagrama de Constelación cuyo diagrama de bloques se muestra en la Figura 3.53.

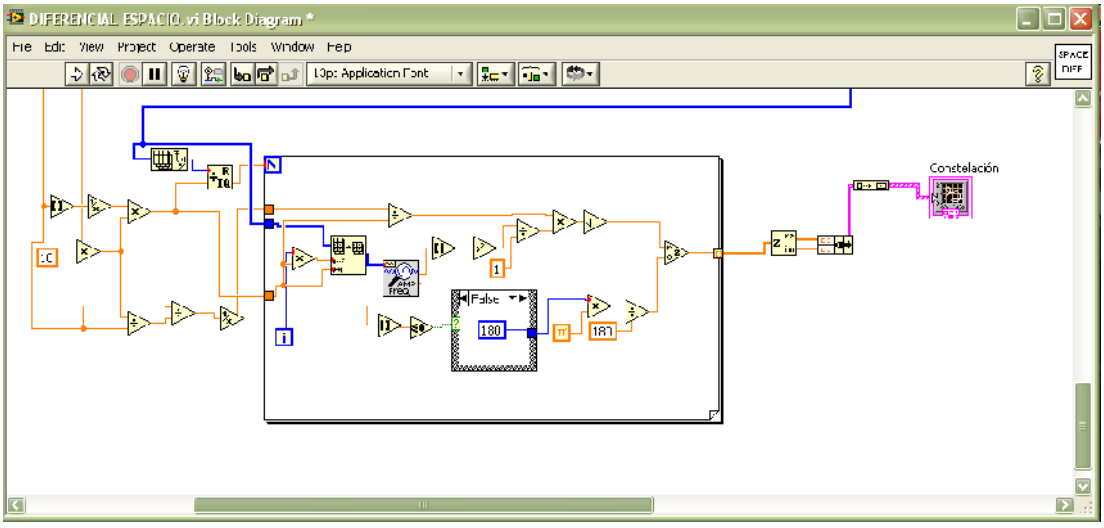


Figura 3.53: Diagrama de Constelación

Para finalizar, se procede a guardar el SubVI desarrollado con el nombre DIFERENCIALESPACIO quedando el **Front Panel** y el **Block Diagram** del SubVI

para el código de línea Diferencial tipo Espacio como se muestra en la Figura 3.54 y la Figura 3.55 respectivamente.

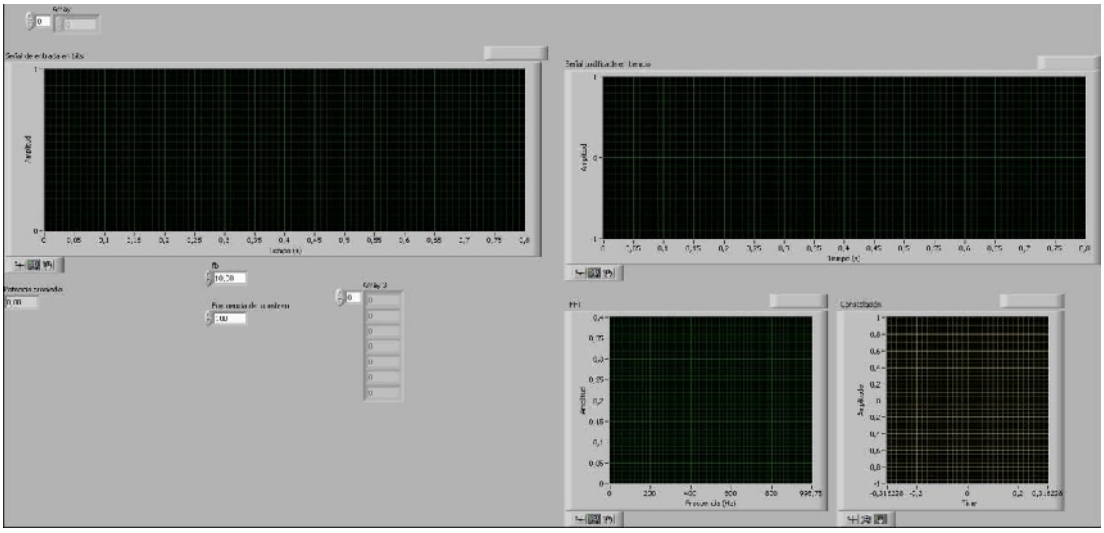


Figura 3.54: Panel Frontal SubVI Código Diferencial Espacio

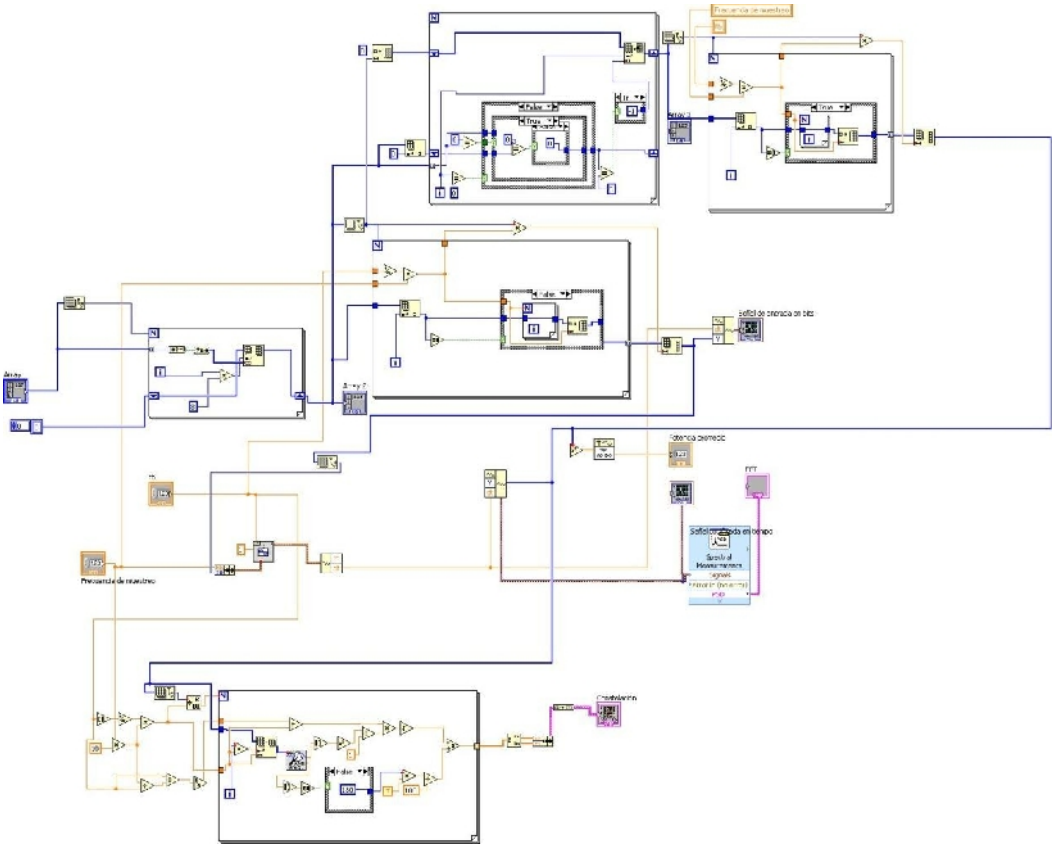


Figura 3.55: Diagrama de Bloques SubVI Código Diferencial Espacio

3.4.4 DESARROLLO SUBVI PARA EL CODIGO DE LINEA DIFERENCIAL MARCA

Para desarrollar este SubVI al igual que para los códigos anteriores es necesario definir los parámetros de entrada tanto en la Front Panel como en el Block Diagram como se indica en la Figura 3.56.

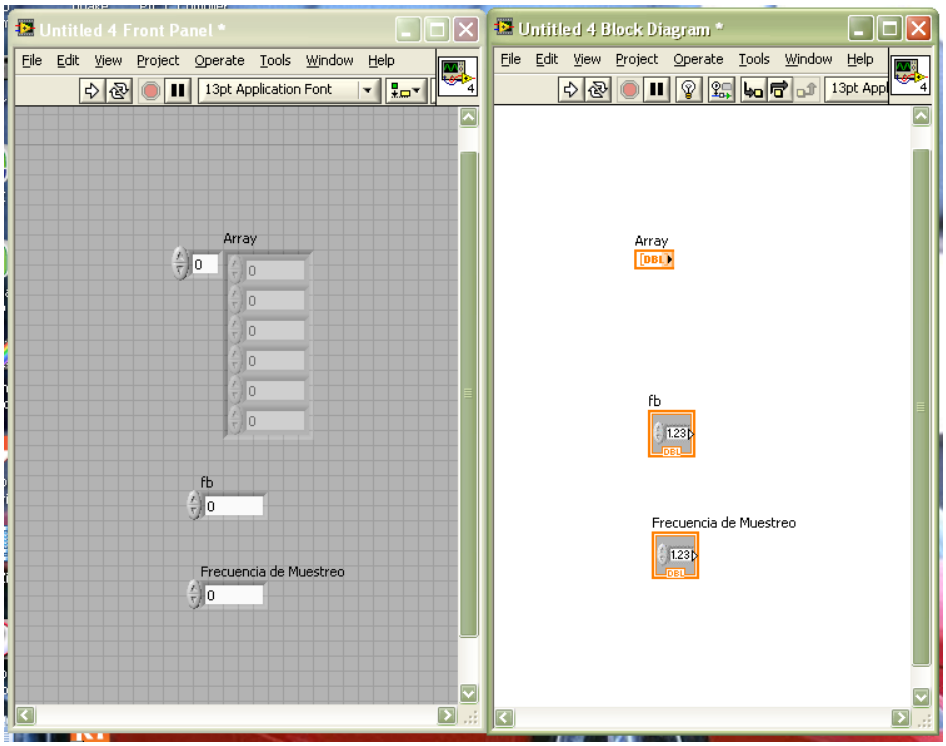


Figura 3.56: Definición de los parámetros iniciales.

Para este código de línea el algoritmo es muy similar al del Diferencial Marca, a diferencia de este, cuando exista el símbolo 1 existirá una transición en los valores de tensión ya sea de positivo a negativo o viceversa dependiendo del estado anterior y cuando exista el símbolo 0 no deberá haber transición por lo que el procedimiento es similar.

Se obtiene el número de elementos para determinar la cantidad de bits que van a ser codificados, este dato se almacena en un segundo arreglo que lleva el título de "Array 2", como se muestra en la Figura 3.57.

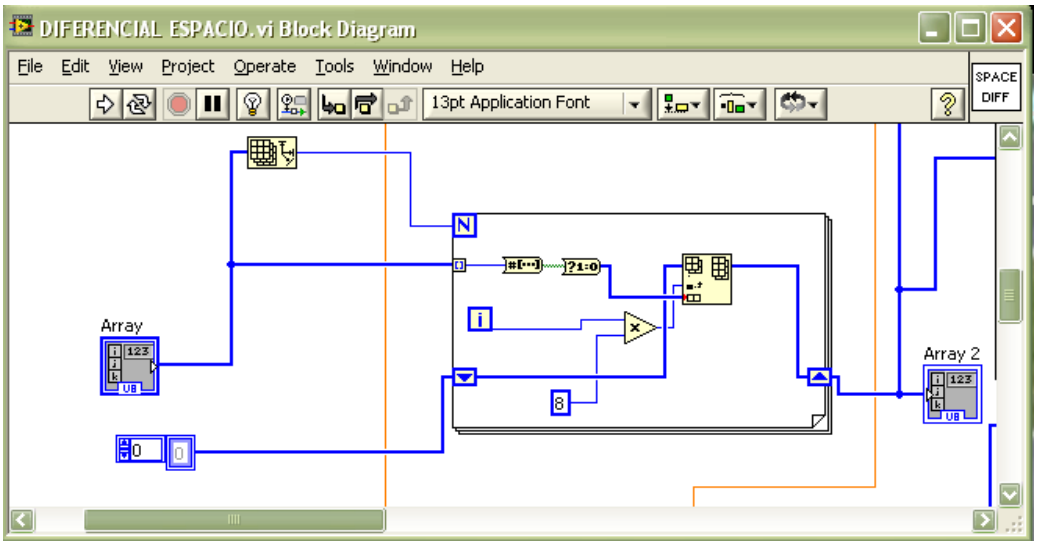


Figura 3.57: Conversión de los parámetros de entrada.

La parte del programa que realiza la codificación de línea Diferencial Espacio es muy similar a la del código Diferencial Marca solo que ahora las transiciones se darán cuando se detecte un uno esto se puede observar en la Figura 3.58, que indica la parte del programa que cumple con el algoritmo de codificación.

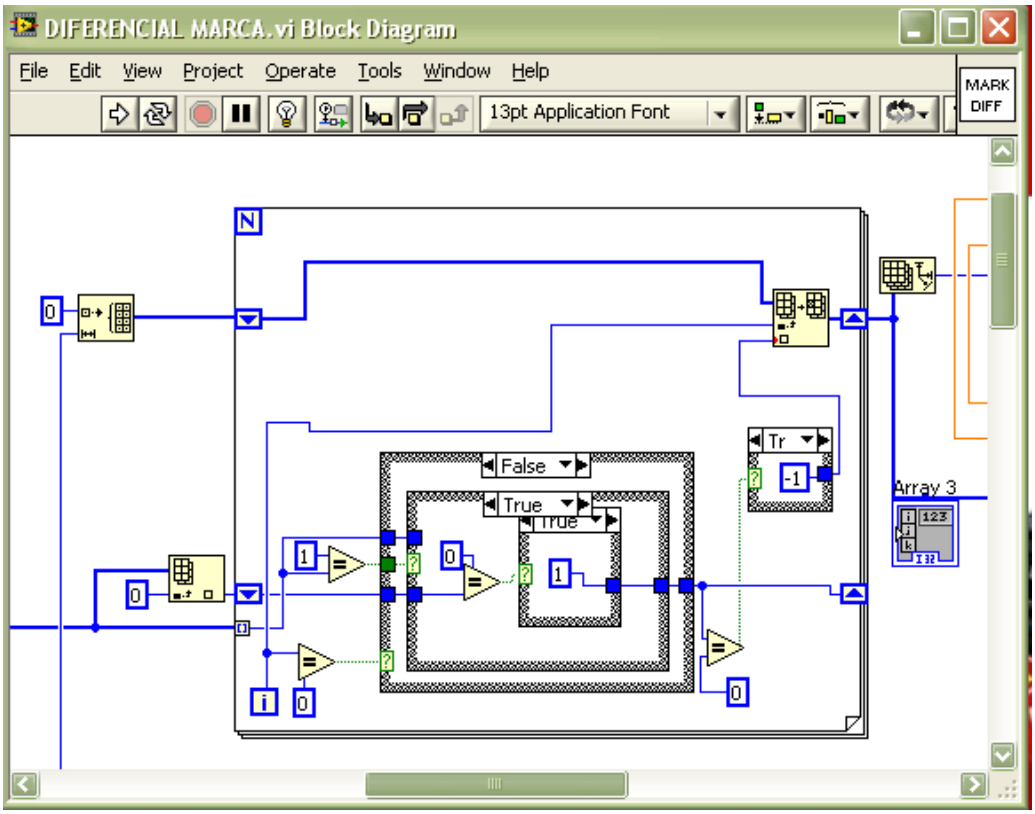


Figura 3.58: Codificación Diferencial Marca.

El procedimiento para visualizar la “Señal de entrada en bits” se observa en la Figura 3.59.

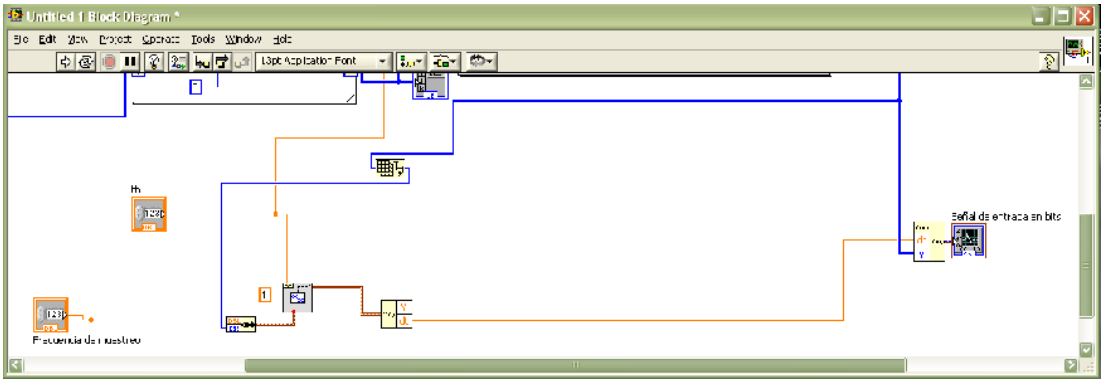


Figura 3.59: Visualización de la Señal d entrada en bits-

Para obtener en el Indicador la Señal Codificada en función del tiempo se sigue el mismo procedimiento que para el Código Diferencial tipo Espacio como se puede verificar en la Figura 3.60.

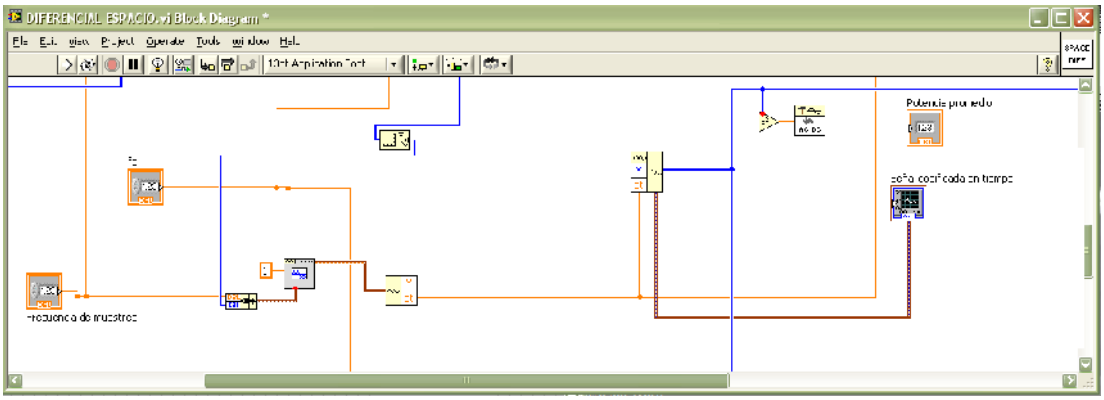


Figura 3.60: Visualización de la señal Codificada en función del tiempo.

Del mismo modo con la ayuda de la función **Spectral Measurements** se realiza la visualización de la Señal Codificada en función de la frecuencia como se muestra en la Figura 3.61.

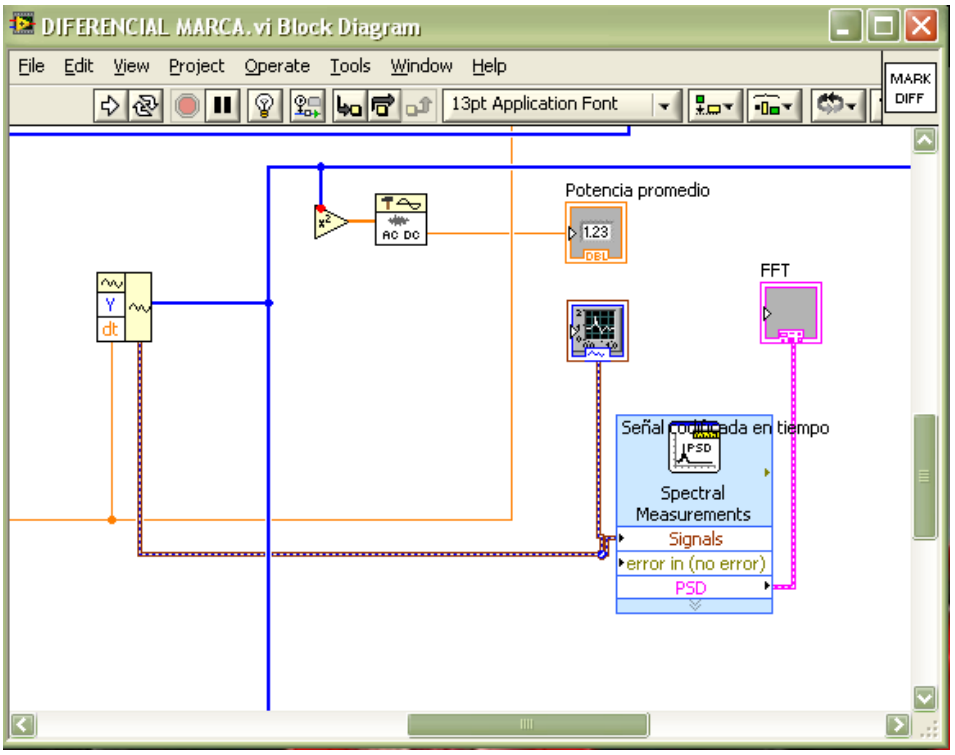


Figura 3.61: Visualización de la señal codificada en función de la frecuencia.

De la misma forma al igual que en todos los casos anterior en este SubVI se ha desarrollado visualización del Diagrama de Constelación cuyas funciones que conforman el diagrama de bloques se muestra en la Figura 3.62.

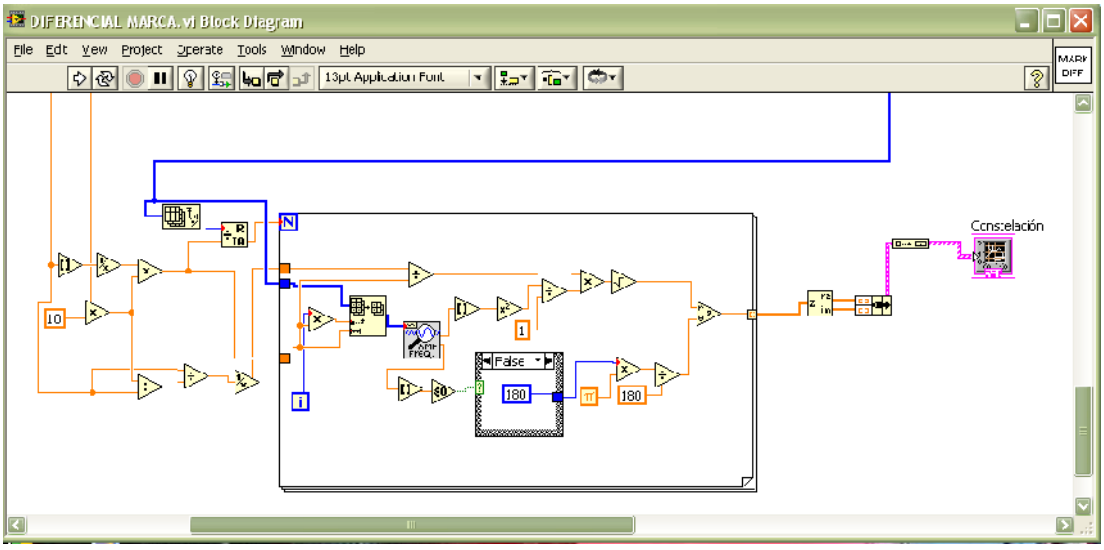


Figura 3.62: Diagrama de Constelación.

A continuación, se procede a guardar el SubVI con el nombre DIFERENCIAL MARCA, quedando el **Front Panel** y el **Block Diagram** del SubVI para el código de

línea Diferencial tipo Marca como se muestra en la Figura 3.63 y la Figura 3.64 respectivamente.

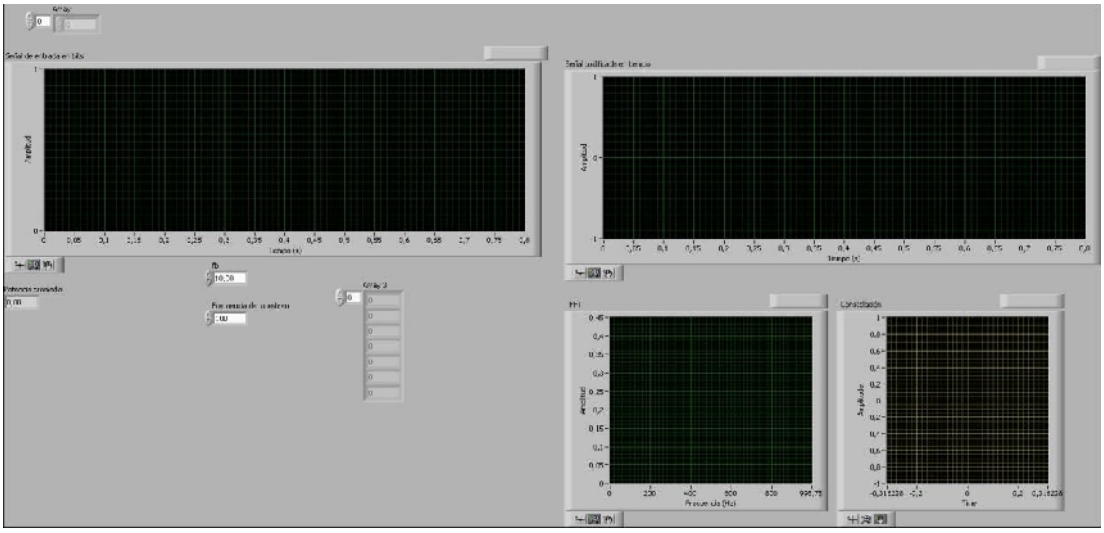


Figura 3.63: Panel Frontal SubVI Código Diferencial Marca

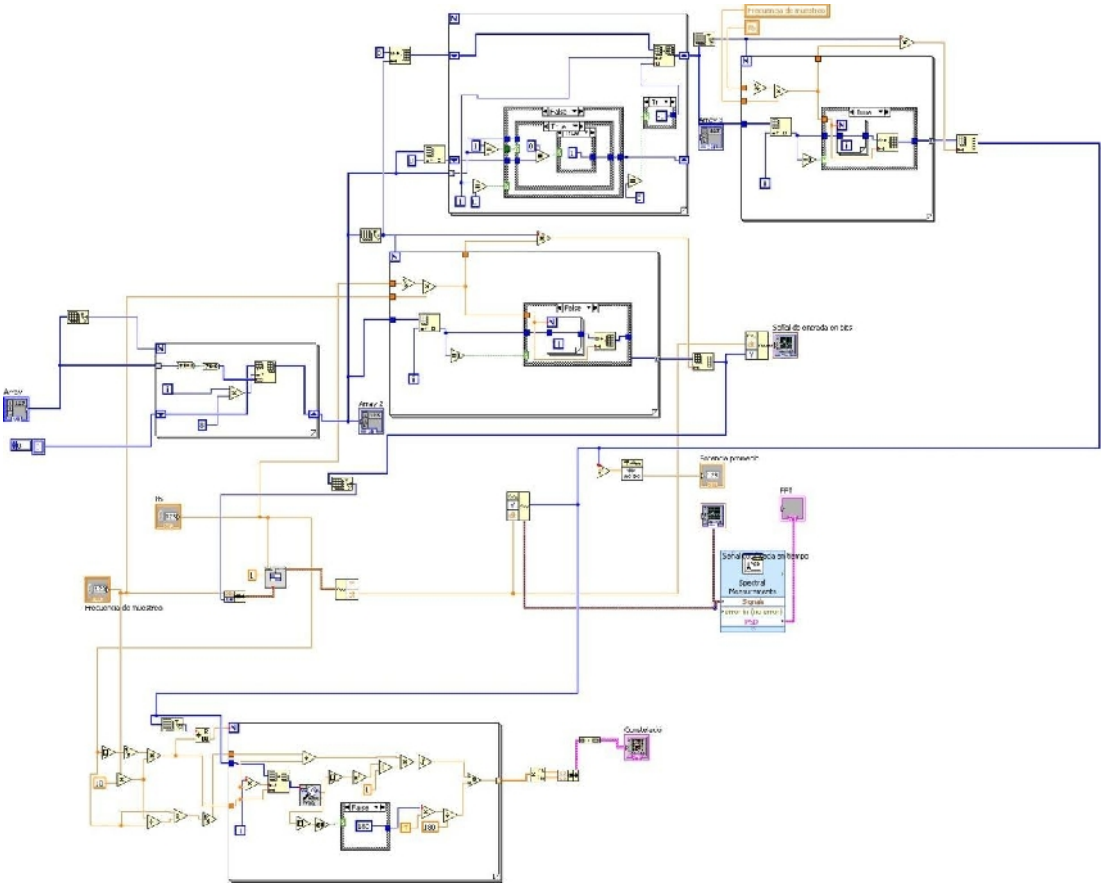


Figura 3.64: Diagrama de Bloques SubVI Código Diferencial Marca.

3.4.5 DESARROLLO SUBVI PARA EL CODIGO DE LINEA AMI TIPO RZ

De la misma forma que para los códigos anteriores se definen los parámetros de entrada como se muestra en la Figura 3.65.

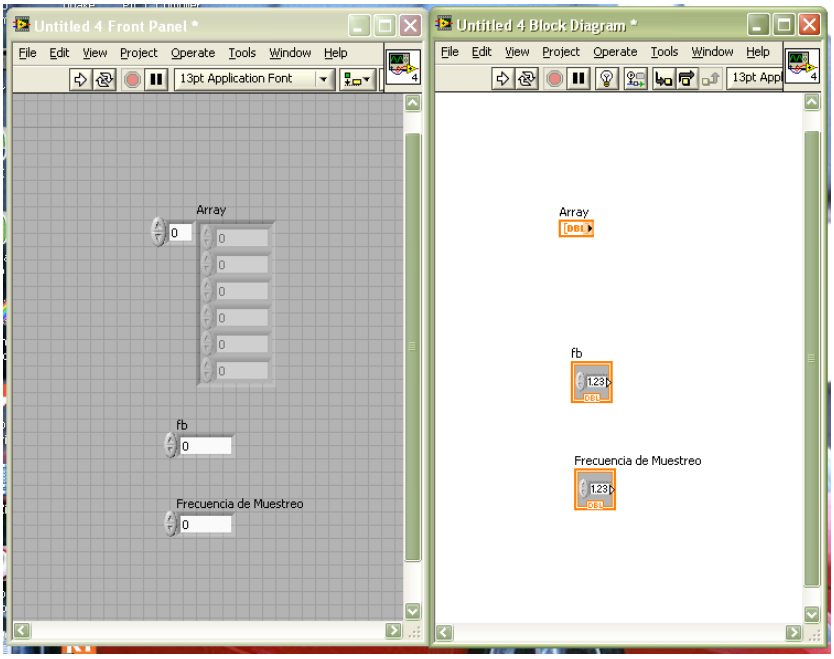


Figura 3.65: Definición de los parámetros de entrada

Ahora se obtienen el número de elementos para determinar la cantidad de bits que van a ser codificados tal como se muestra en la Figura 3.66.

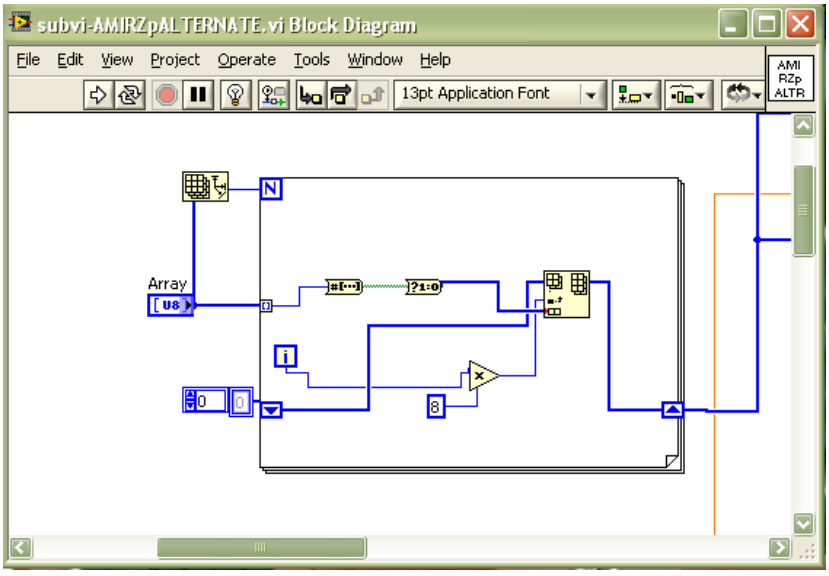


Figura 3.66: Conversión de los datos de entrada.

Continuando con el algoritmo de codificación del código de línea AMI del tipo RZ para el cual se representa al símbolo “1” con valores de voltaje positivos y negativos de forma alternada y produciendo el paso por el nivel cero a la mitad del intervalo esta parte del algoritmo se muestra en la Figura 3.67 y Figura 3.68, para lo cual en la función **Case Structure** que se encuentra dentro del **For Loop**, en esta parte del programa se ubican los niveles positivos y negativos de forma alternada para cada 0 del o los datos ingresados.

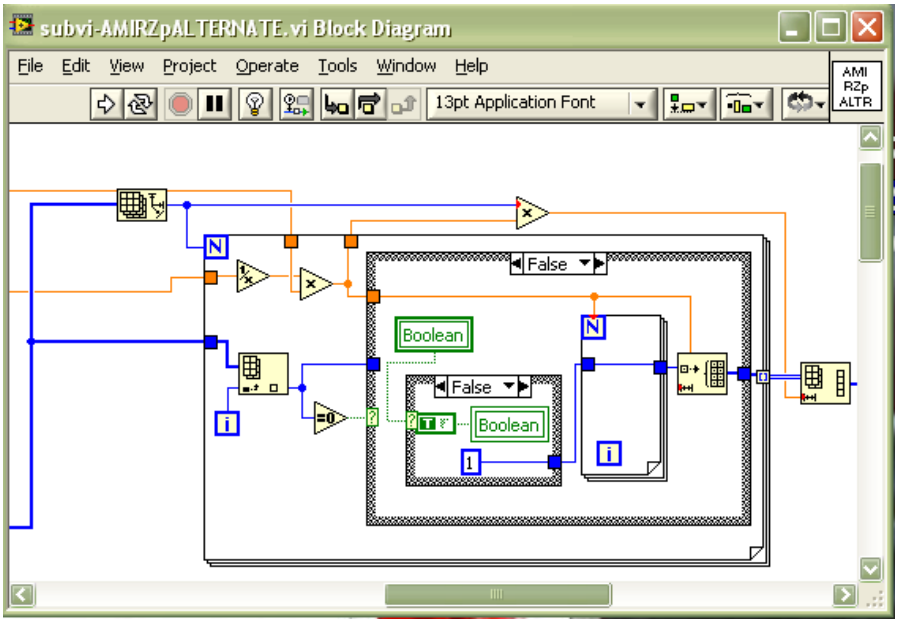


Figura 3.67: Codificación AMI RZ Niveles Positivos para el símbolo 0.

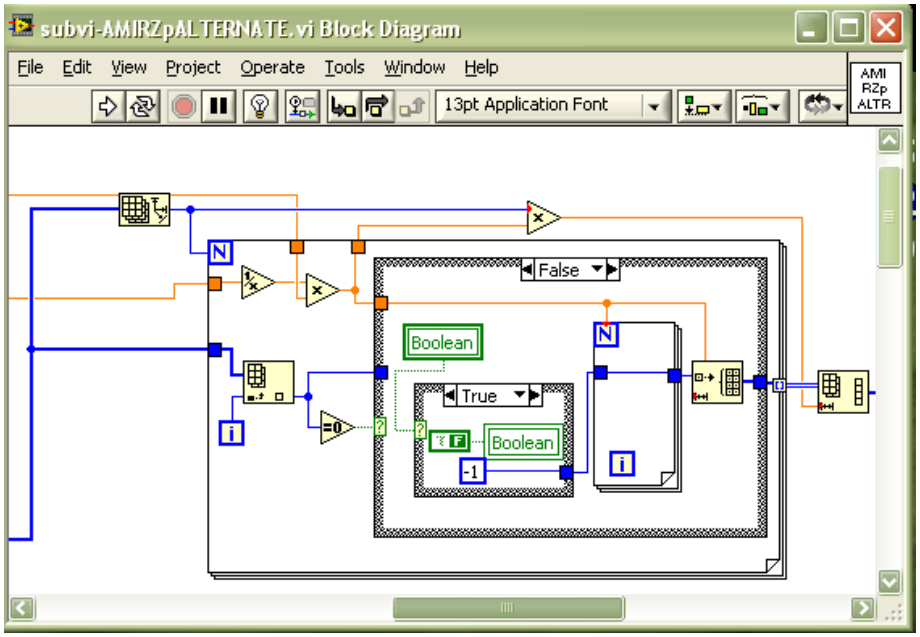


Figura 3.68: Codificación AMI RZ Niveles Negativos para el símbolo 0.

El segmento del programa que realiza la parte del algoritmo correspondiente al símbolo “0” el cual tomara el nivel 0 se muestra en la Figura 3.69.

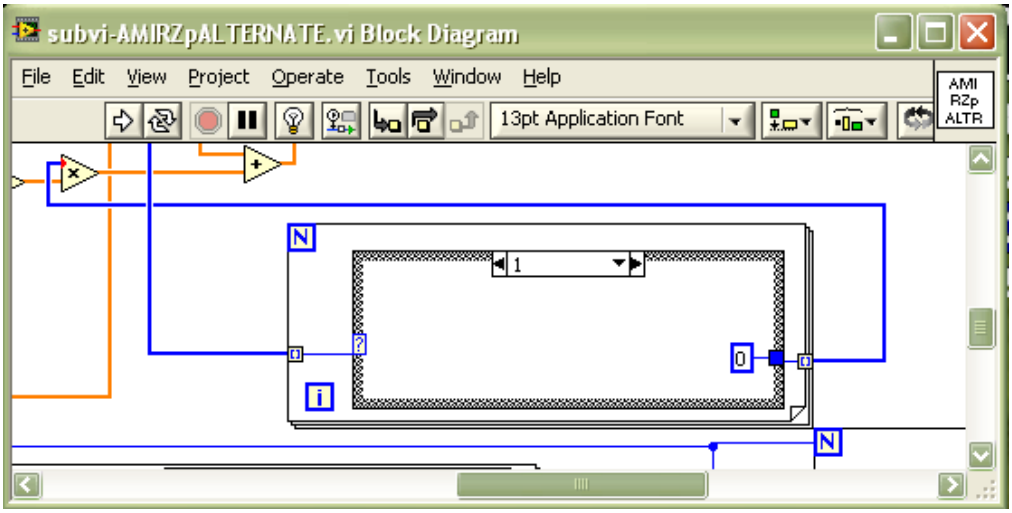


Figura 3.69: Codificación AMI RZ para el símbolo 1.

Para visualizar en el **Front Panel** mediante Indicador que lleva por título “Señal de entrada en bits” es necesario indicar los parámetros que va a tener la nueva onda en base a la frecuencia de muestreo la de bit y el número de bits, la parte del programa que realiza esta función se la muestra en la Figura 3.70.

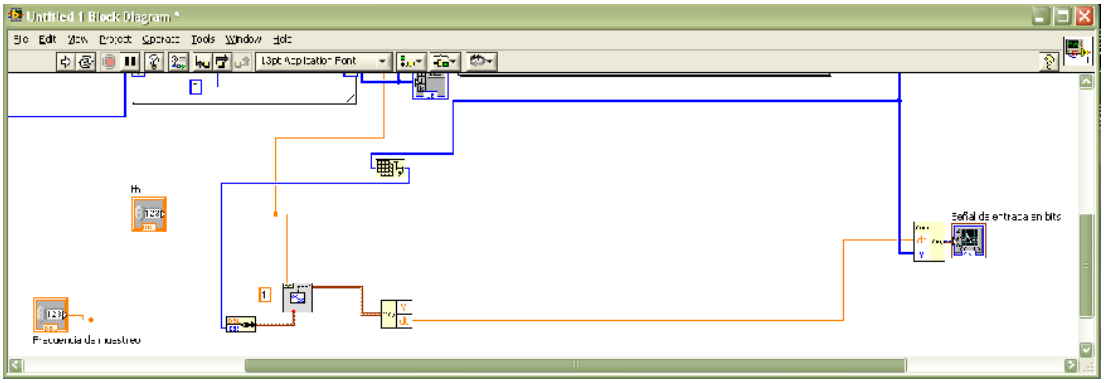


Figura 3 70: Visualización de la Señal de entrada en bits.

Con estos parámetros se obtiene la visualización de la señal codificada en función del tiempo y de la frecuencia como se indica en la Figura 3.71.

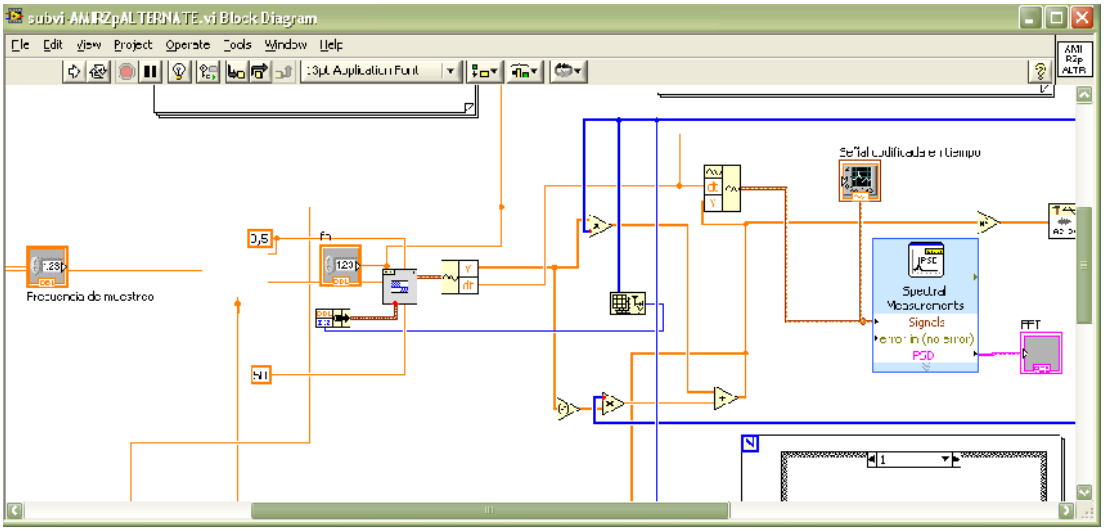


Figura 3.71: Visualización de la señal codificada en función del tiempo y la frecuencia.

Continuando de la misma forma que en los códigos anterior se ha desarrollado visualización del Diagrama de Constelación en este SubVI y las funciones que permiten que esta visualización sea posible se muestran en la Figura 3.72.

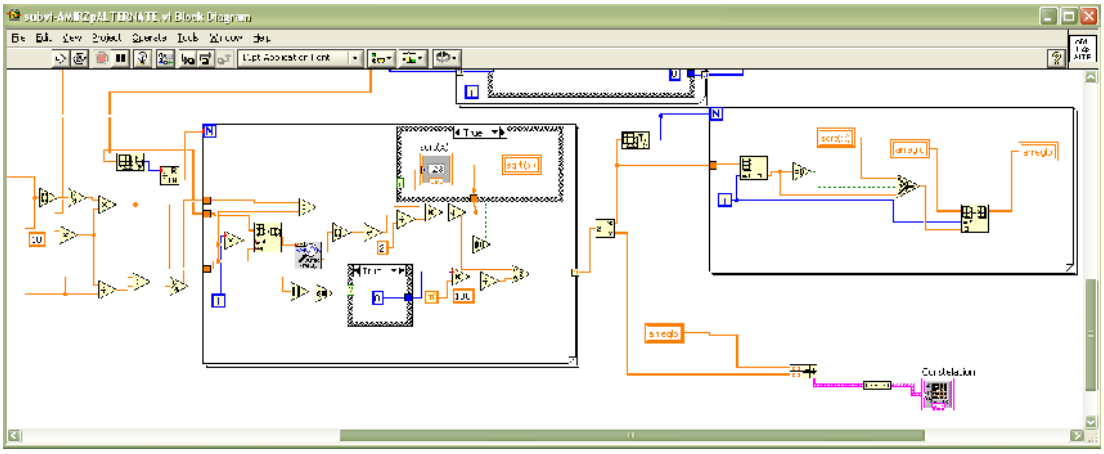


Figura 3.72: Diagrama de Constelación.

Para terminar esta parte del programa se procede a guardar el SubVI con el nombre de AMIRZp, quedando el **Front Panel** y el **Block Diagram** del SubVI para el Código de Línea AMI normal con retorno a cero como se muestra en la Figura 3.73 y la Figura 3.74 respectivamente.

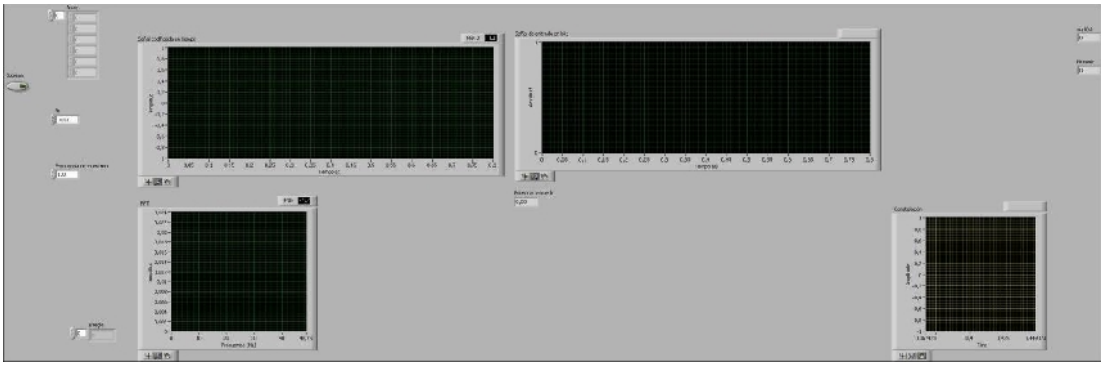


Figura 3.73: Panel Frontal SubVI Código AMI RZ

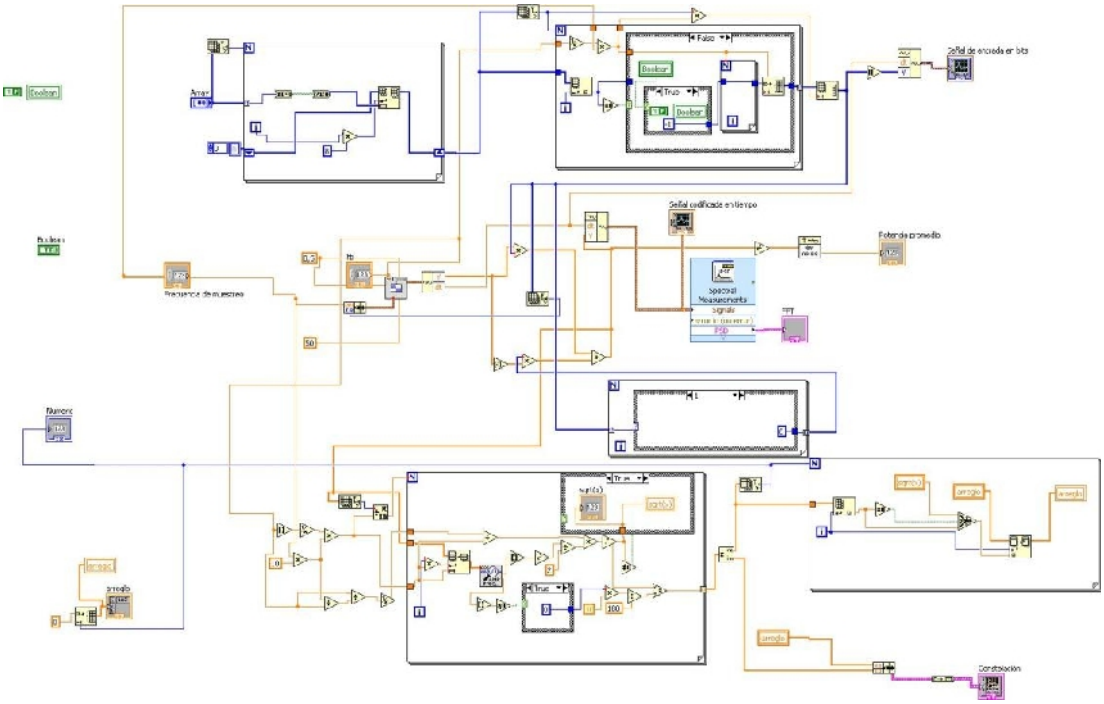


Figura 3.74: Diagrama de Bloques SubVI Código AMI RZ

3.4.6 DESARROLLO SUBVI PARA EL CODIGO DE LINEA AMI TIPO NRZ

Al igual que se ha realizado para todos los anteriores programas es necesario definir los parámetros de inicio para este SubVI como se puede observar en la Figura 3.75.

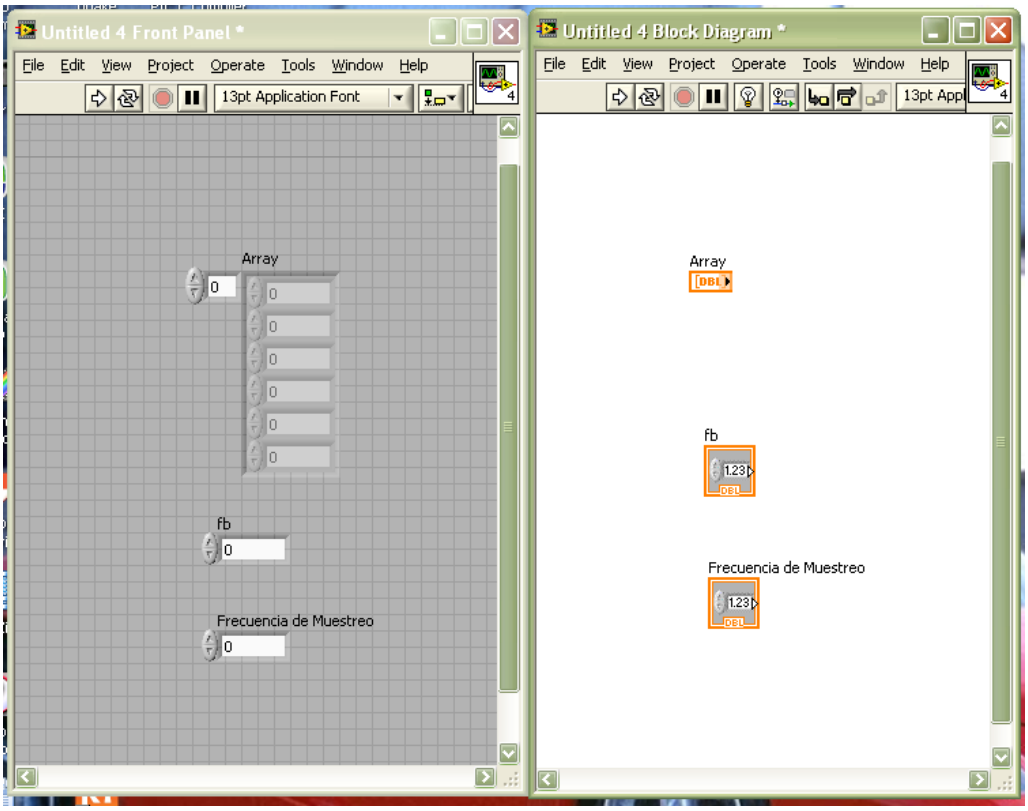


Figura 3.75: Definición de los parámetros iniciales.

Del arreglo inicial se obtiene el número de bits, esto se realiza con las funciones que se muestran en la Figura 3.76.

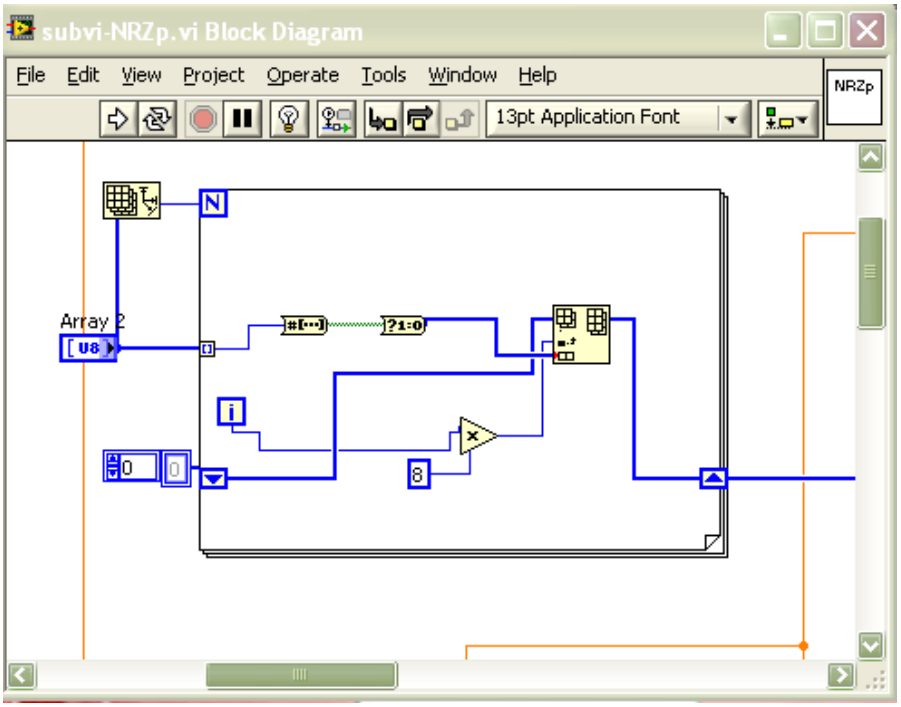


Figura 3.76: Conversión de los datos de entrada

Continuando con el algoritmo de codificación del código de línea AMI del tipo NRZ para el cual se representa al símbolo “1” con valores de voltaje positivos y negativos de forma alternada sin pasar por el nivel cero a la mitad del intervalo esta parte del algoritmo se la observa en la Figura 3.77, para lo cual en la función **Case Structure** que se encuentra del **For Loop**, aquí vamos a ubicar los niveles positivos y negativos de forma alternada para cada 0 que ingrese como dato.

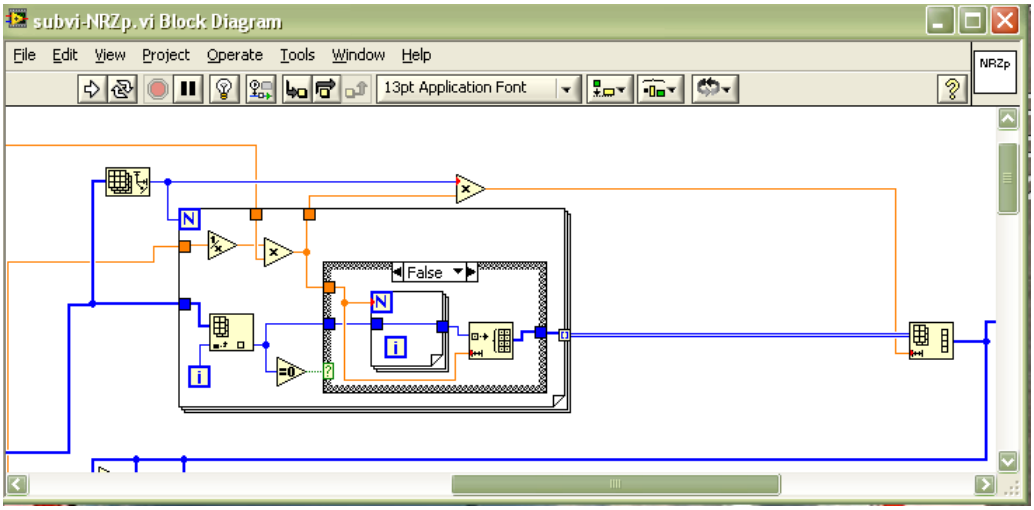


Figura 3.77: Codificación de Línea para el símbolo 0-

Ahora para asignar el valor de cero para el símbolo “0”, se puede observar en la Figura 3.78 el segmento del programa que realiza esta parte de la codificación

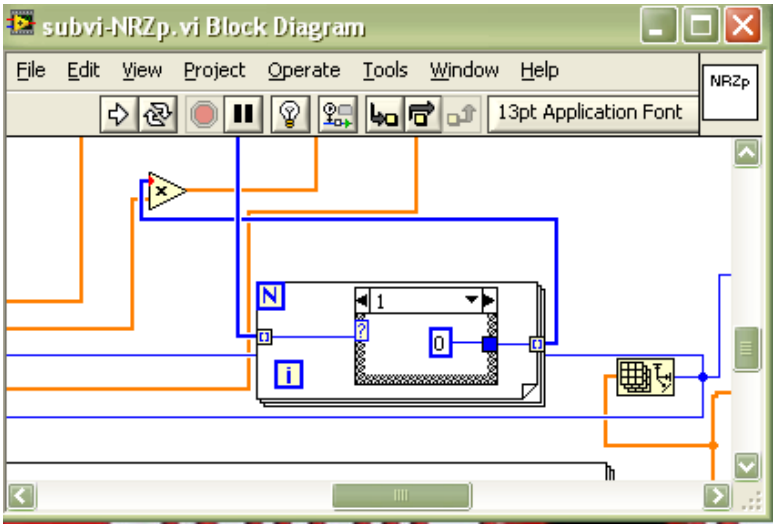


Figura 3.78: Codificación de línea para el símbolo 1.

Para la visualización en el **Front Panel** mediante Indicador que tiene el título "Señal de entrada en bits" es necesario indicar los parámetros de la nueva onda en base a la frecuencia de muestreo la de bit y el numero de bits, esto se lo consigue con la ayuda de la función **Square Waveform,** la parte del programa que realiza esta función se la muestra en la Figura 3.79.

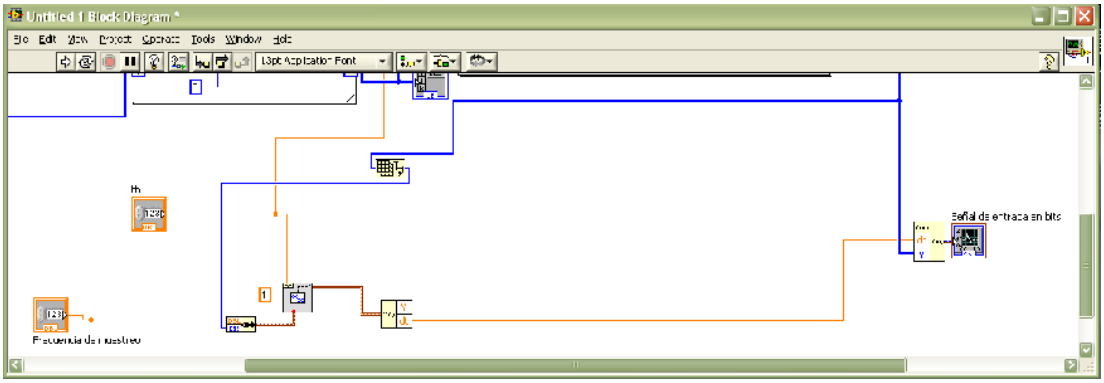


Figura 3.79: Visualización para la señal de entrada en bits.

Continuando con el programa y con los parámetros obtenidos se logra la visualización de la señal codificada en función del tiempo y de la frecuencia como se indica en la Figura 3.80.

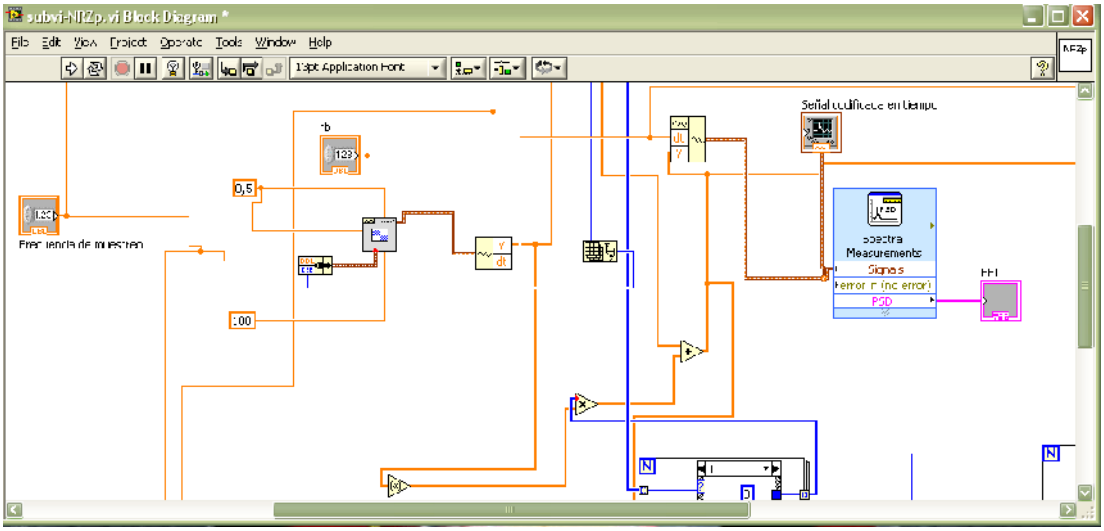


Figura 3.80: Visualización de la señal codificada en función del tiempo y frecuencia.

A continuación en la Figura 3.81, se puede observar cómo está desarrollado el programa para poder obtener la visualización del Diagrama de Constelación de este SubVI.

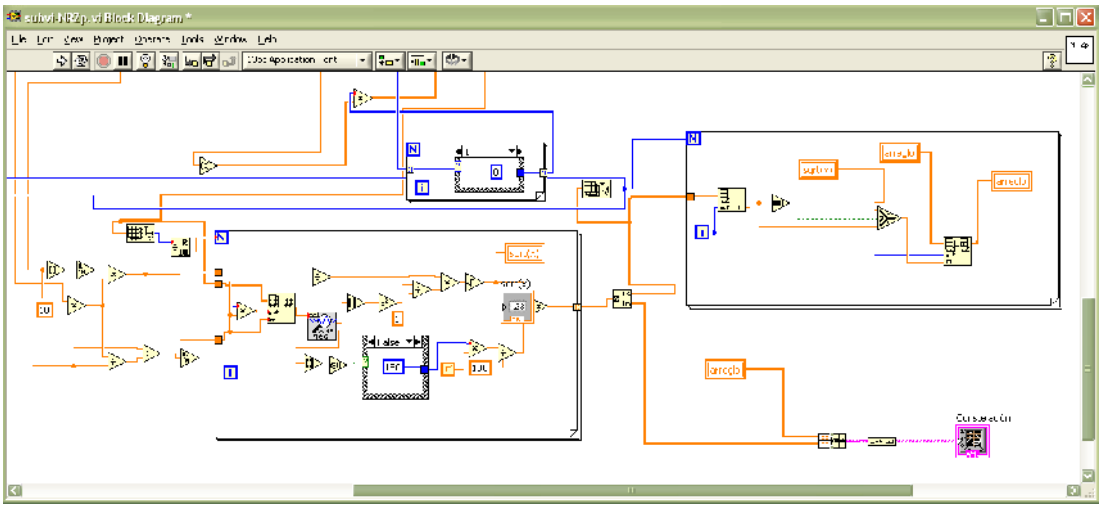


Figura 3.81: Visualización del Diagrama de Constelación

Para concluir se guarda el SubVI en este caso con el nombre de subvi-NRZp quedando el **Front Panel** y el **Block Diagram** del SubVI para el código de línea AMI normal sin retorno a cero como se muestra en la Figura 3.82 y la Figura 3.83 respectivamente.

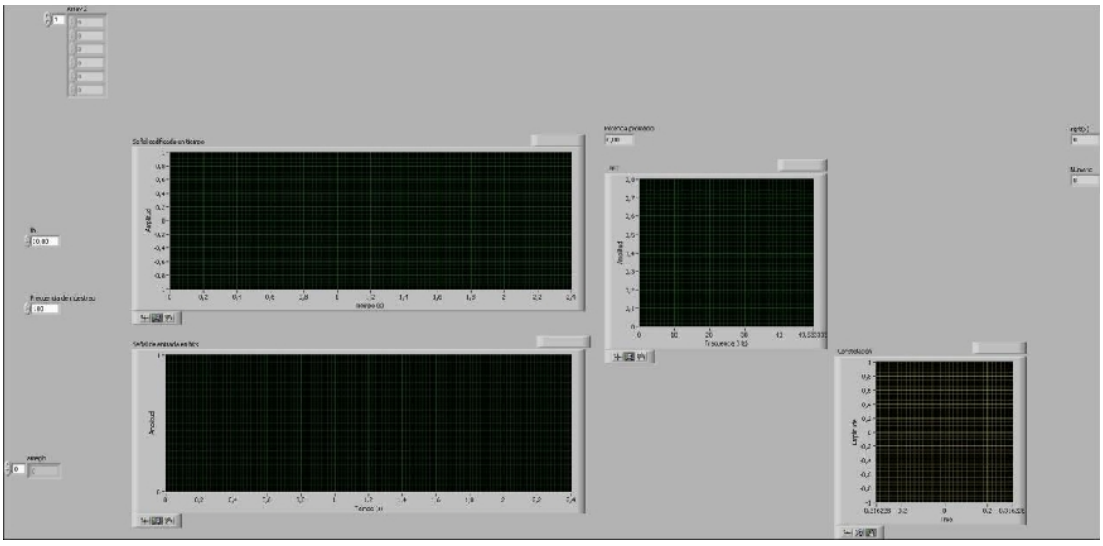


Figura 3.82: Panel Frontal SubVI Código AMI NRZ

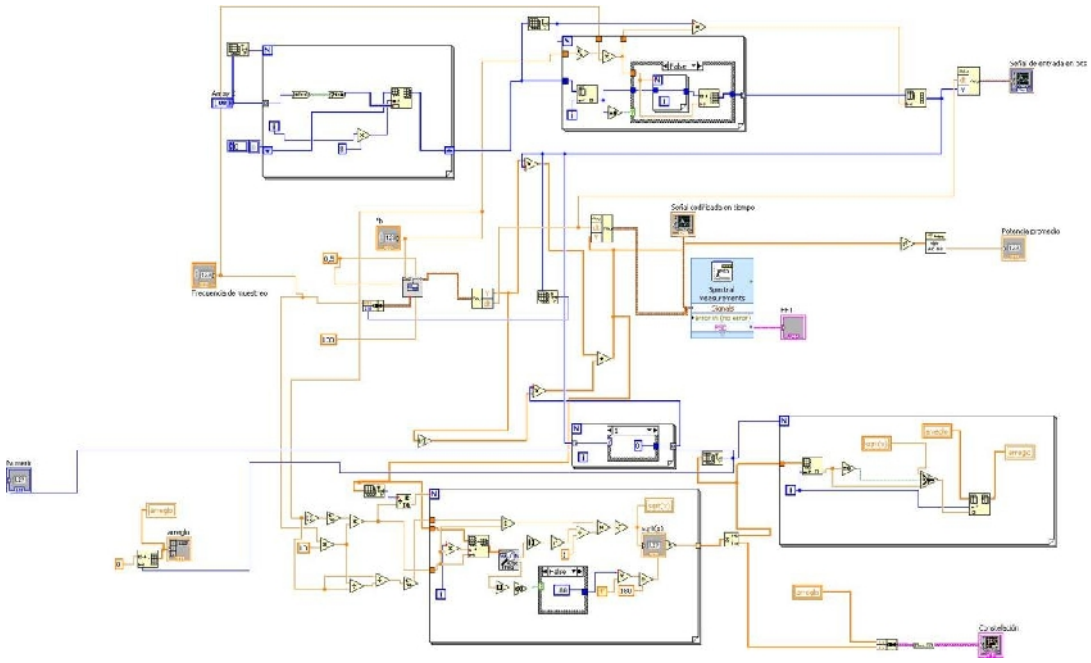


Figura 3.83: Diagrama de Bloques SubVI Código AMI NRZ

3.4.7 DESARROLLO SUBVI PARA EL CODIGO DE LINEA MANCHESTER TIPO L

Al igual que para los casos anteriores se debe de indicar los parámetros de entrada tanto en el **Front Panel** como en el **Block Diagram**, como se puede visualizar en la Figura 3.84.

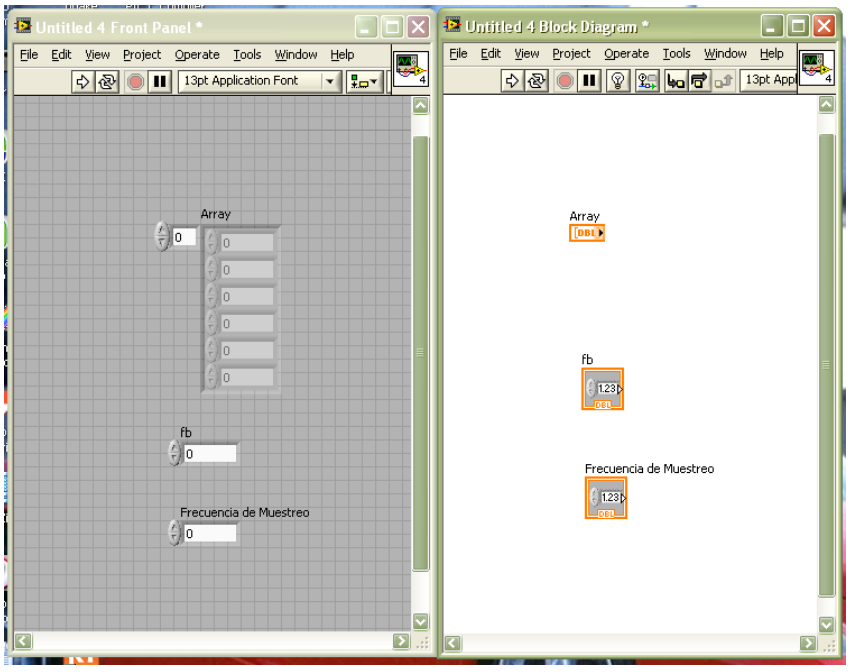


Figura 3.84: Definición de los parámetros de entrada.

Seguido se obtiene el número de símbolos que forman parte del dato a ser codificado, y se lo almacena en la función denominada “Array 2” como se observa en la Figura 3.85.

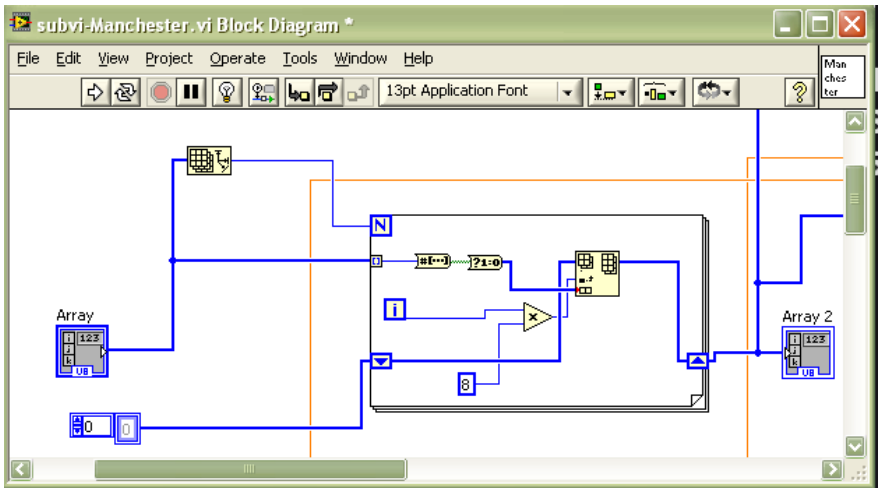


Figura 3.85: Conversión de los parámetros de entrada

La parte del programa que se encarga de cumplir con el algoritmo del código Manchester tipo L se muestra en la Figura 3.86, el cual indica que cada bit que se codifica tiene una transición a la mitad del tiempo de bit y una transición de negativo a positivo representara un 0 y una transición de positivo a negativo representara un 1.

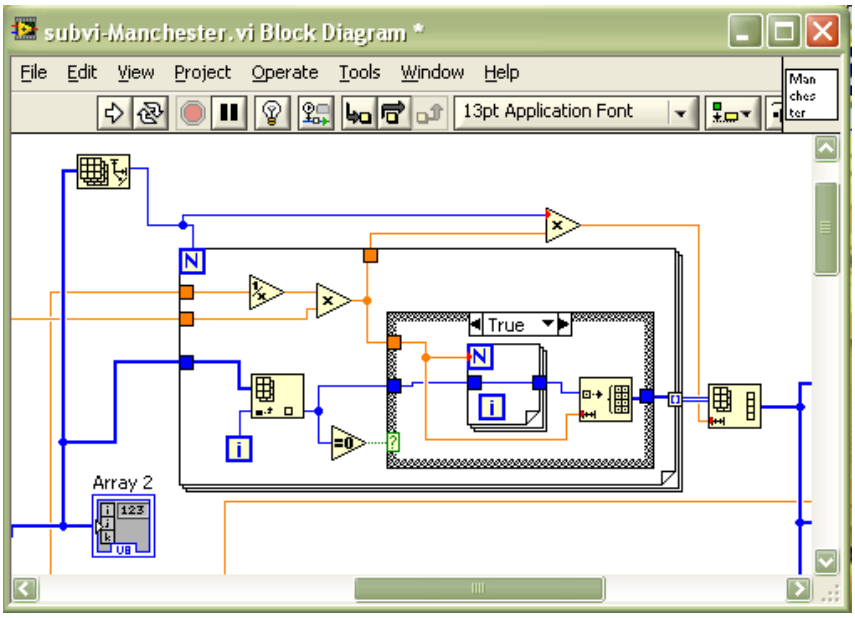


Figura 3.86: Codificación Manchester tipo L.

Ahora se procede a crear las funciones que permitirán visualizar en el **Front Panel** la forma de onda de la señal de entrada mediante el Indicador que tiene el título "Señal de entrada en bits" para lo cual es necesario indicar los parámetros que va a tener nuestra nueva onda en base a la frecuencia de muestreo la de bit y el numero de bits con la ayuda de la función **Square Waveform,** esta parte del programa se la muestra en la Figura 3.87.

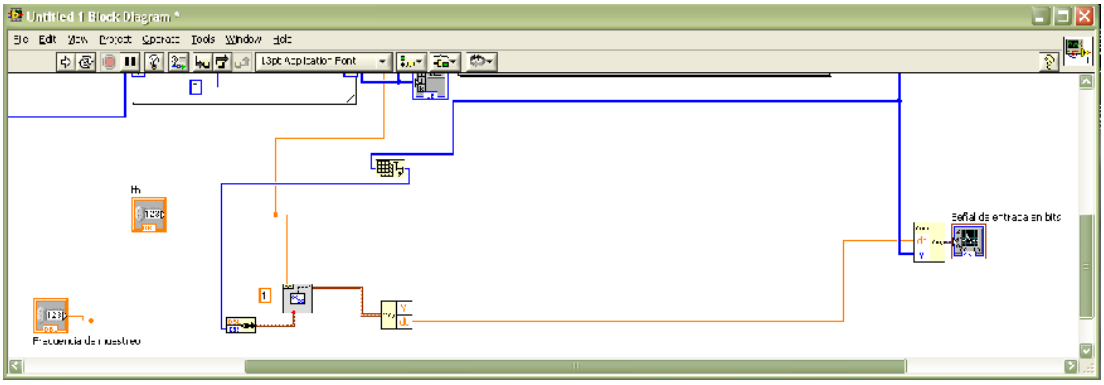


Figura 3.87: Visualización de la señal d entrada en bits.

Una vez obtenidos estos parámetros se procede al desarrollo de la visualización de la señal codificada en función del tiempo y de la frecuencia tal como se indica en la Figura 3.88.

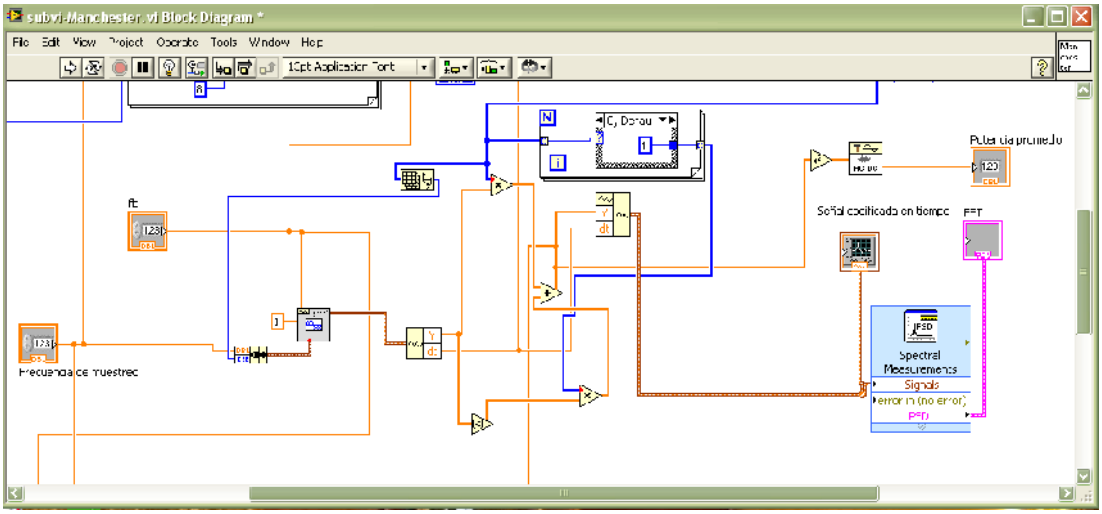


Figura 3.88: Visualización de la señal codificada en tiempo y frecuencia.

A continuación en la Figura 3.89, se observa las funciones utilizadas para obtener el Diagrama de Constelación para el código de línea Manchester tipo L.

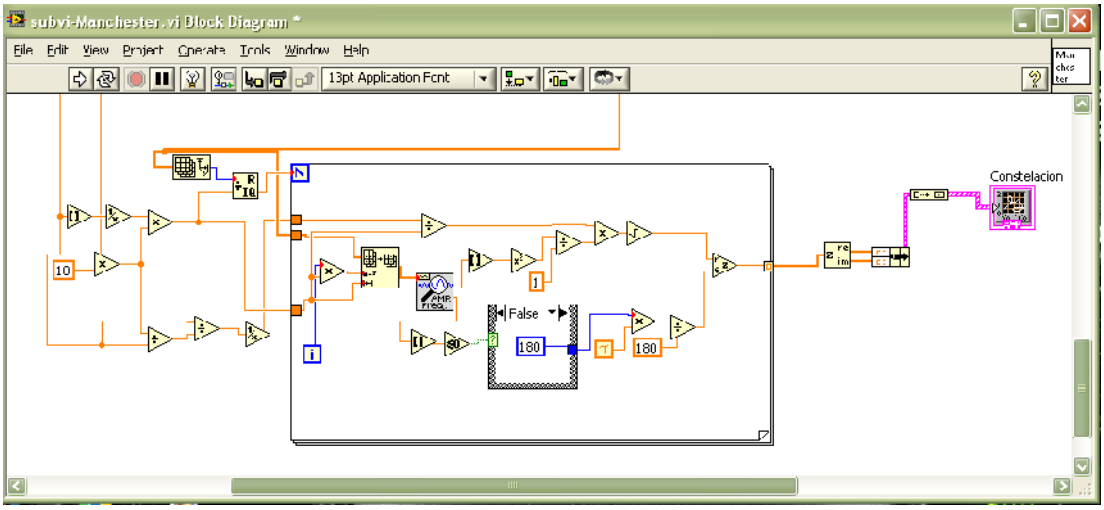


Figura 3.89: Visualización del Diagrama de Constelación.

Para finalizar se procede a guardar el SubVI con el nombre de subvi-Manchester quedando el **Front Panel** y el **Block Diagram** del SubVI para el código de línea Manchester tipo L como se muestra en la Figura 3.90 y la Figura 3.91 respectivamente.

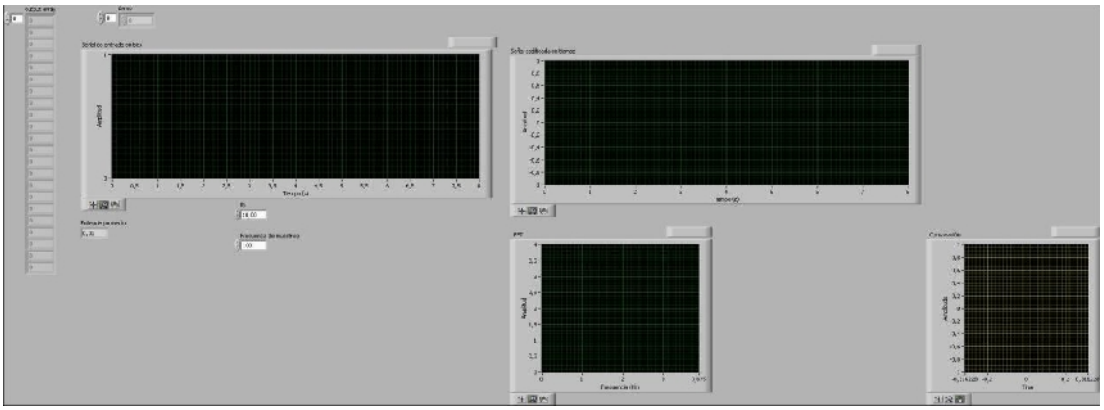


Figura 3.90: Panel Frontal SubVI Código Manchester tipo L.

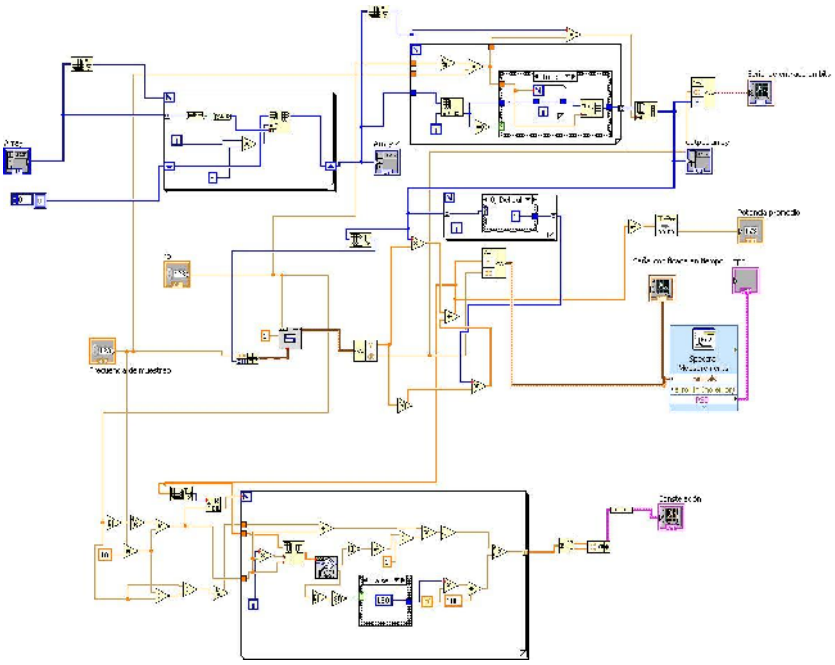


Figura 3.91: Diagrama de Bloques SubVI Código Manchester tipo L.

3.4.8 DESARROLLO SUBVI PARA EL CODIGO DE LINEA MANCHESTER DIFERENCIAL

Al igual que se ha realizado anteriormente se definen los parámetros de inicio para este SubVI como se puede observar en la Figura 3.92.

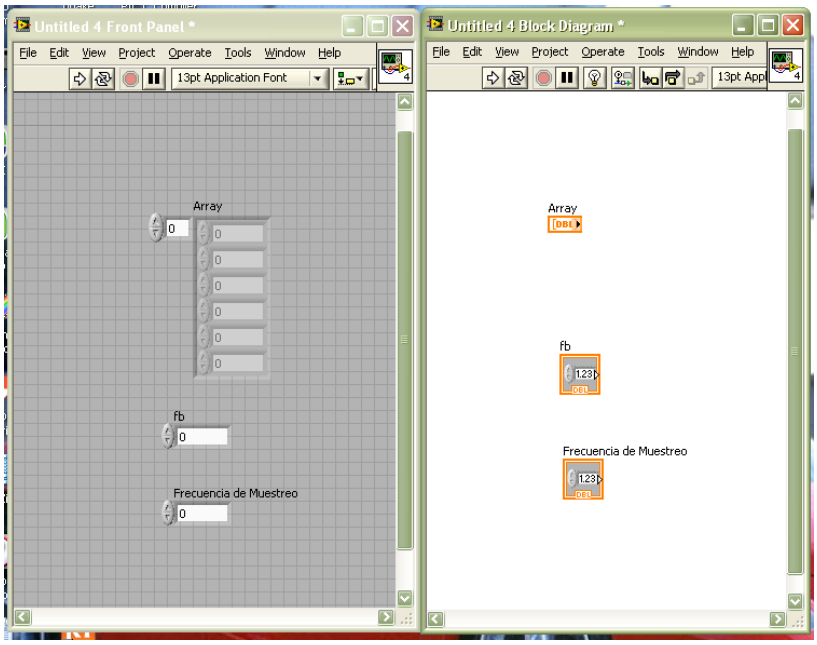


Figura 3.92: Definición de los parámetros de entrada.

Luego se obtiene el número de símbolos a ser codificados, almacenándolos en una función denominada “Array 2” como se observa en la Figura 3.93.

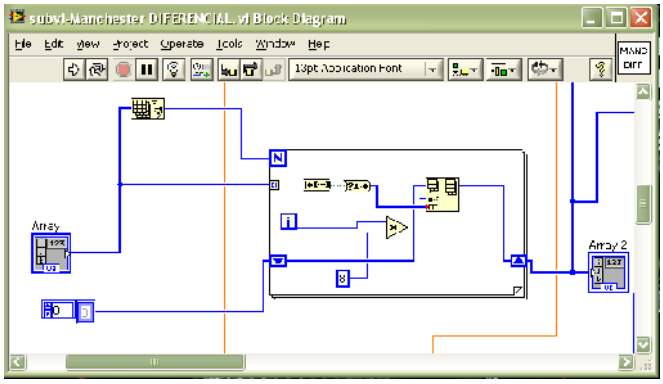


Figura 3.93: Conversión de los parámetros de entrada.

Para que el programa cumpla con el algoritmo de codificación del código de línea Manchester Diferencial el cual indica que el símbolo '1' se lo indica haciendo que la primera mitad de la señal sea igual a la última mitad del bit anterior, es decir no existe transición al inicio del intervalo mientras que un bit '0' se lo indica haciendo que en la primera mitad de la señal sea contraria a la última mitad del último bit, es decir, con una transición al principio del bit

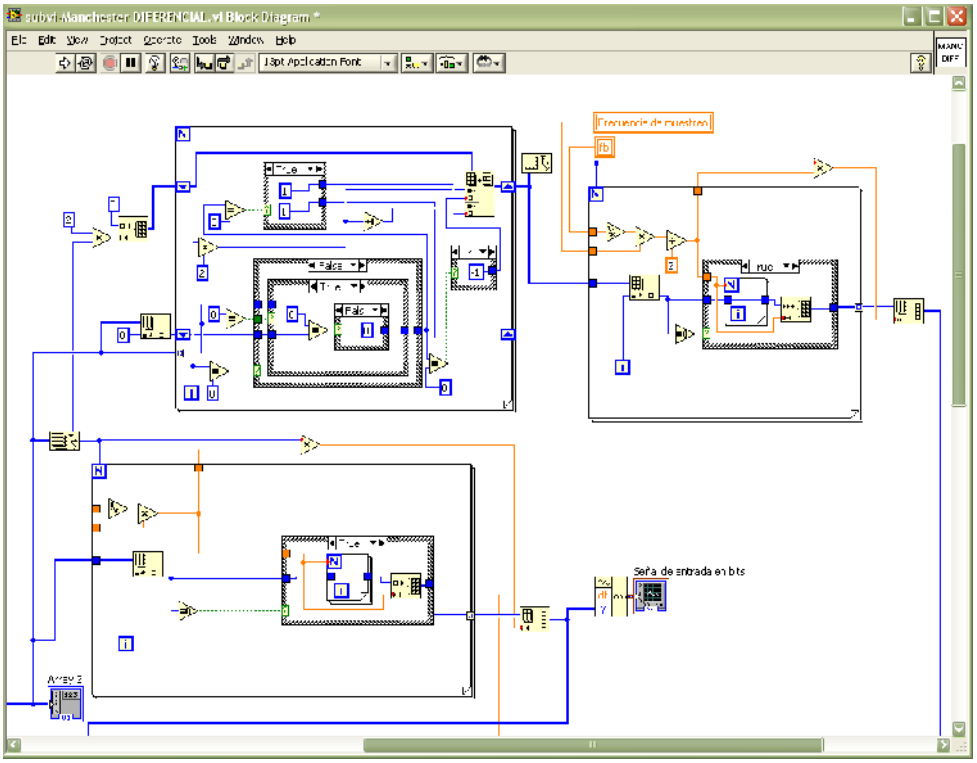


Figura 3.94: Codificación Manchester Diferencial

Para visualizar en el **Front Panel** la onda que representa a la "Señal de entrada en bits" se necesitan indicar los parámetros de la nueva onda en base a la frecuencia de muestreo la de bit y el numero de bits que serán los parámetros a ser ingresados en la función **Square Waveform,** esta parte del programa se muestra en la Figura 3.95.

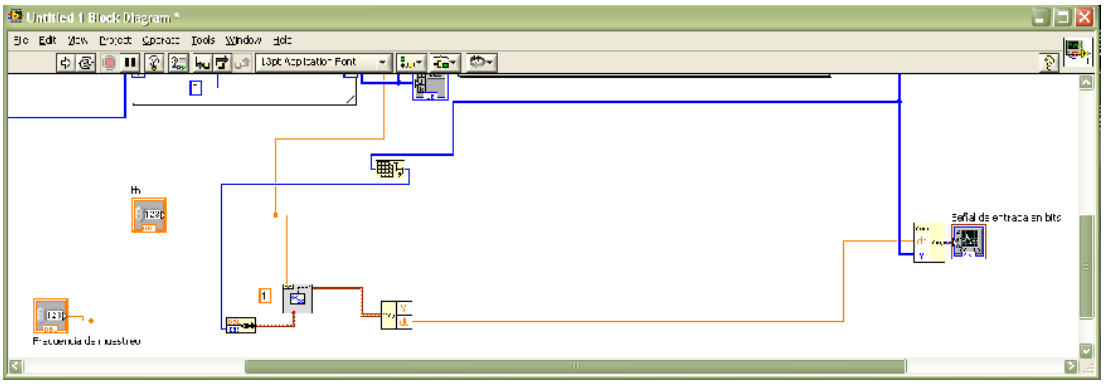


Figura 3.95: Visualización de la señal de entrada en bits.

Una vez obtenidos estos parámetros se procede con el desarrollo de la gráfica para la visualización de la señal codificada en función del tiempo y de la frecuencia como se puede ver en la Figura 3.96.

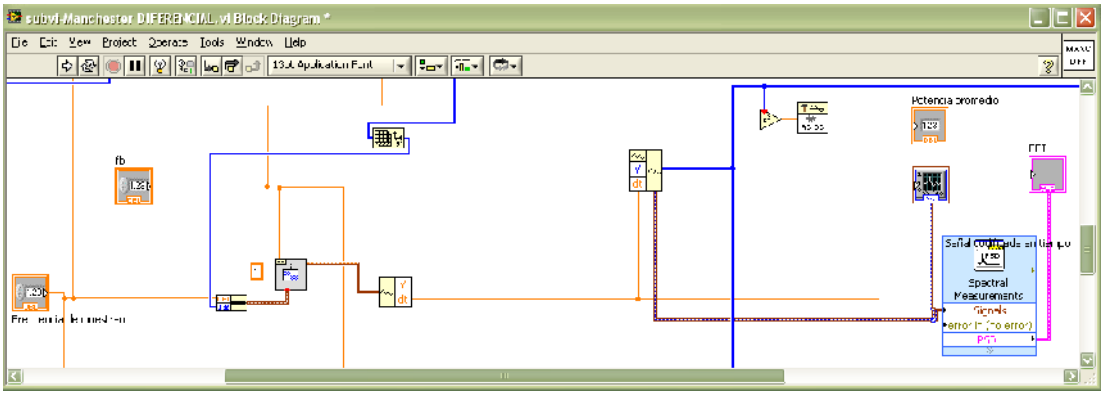


Figura 3.96: Visualización de la señal codificada en función del tiempo y la frecuencia.

De igual forma que para los códigos desarrollados anteriormente en la Figura 3.97 se puede visualizar las funciones utilizadas para obtener el Diagrama de Constelación para el código de línea Manchester Diferencial.

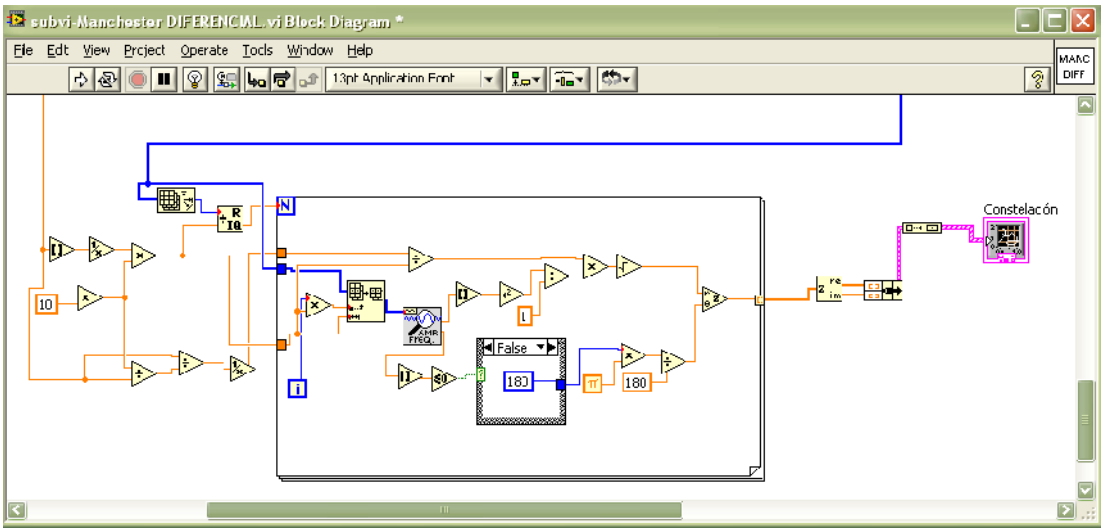


Figura 3.97: Visualización del Diagrama de Constelación.

Para concluir con el presente programa se guarda el SubVI para este caso si utilizó el nombre de subvi-Manchester DIFERENCIAL quedando el **Front Panel** y el **Block Diagram** del SubVI para el código de línea Manchester Diferencial como se muestra en la Figura 3.98 y la Figura 3.99 respectivamente.

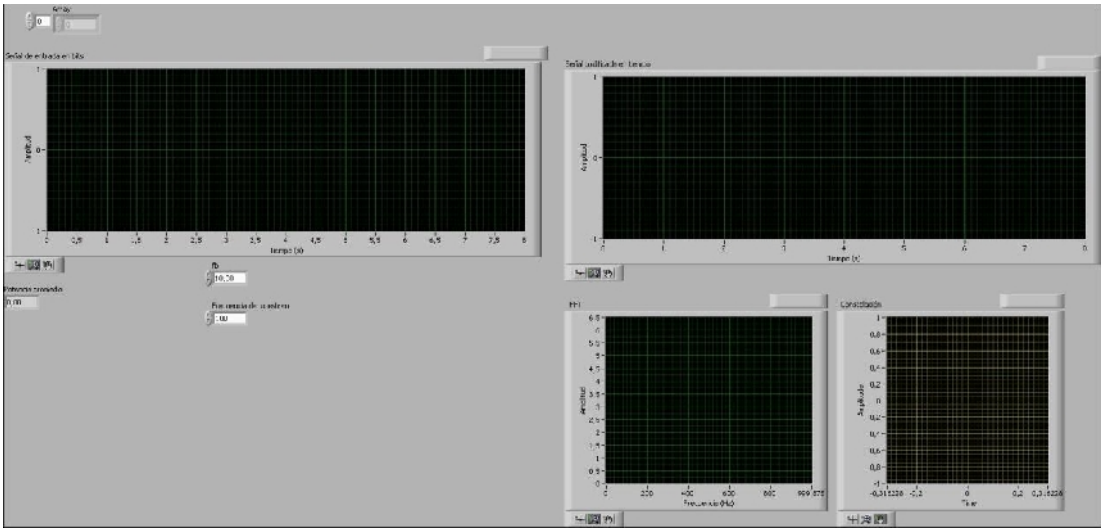


Figura 3.98: Panel Frontal SubVI Código Manchester Diferencial.

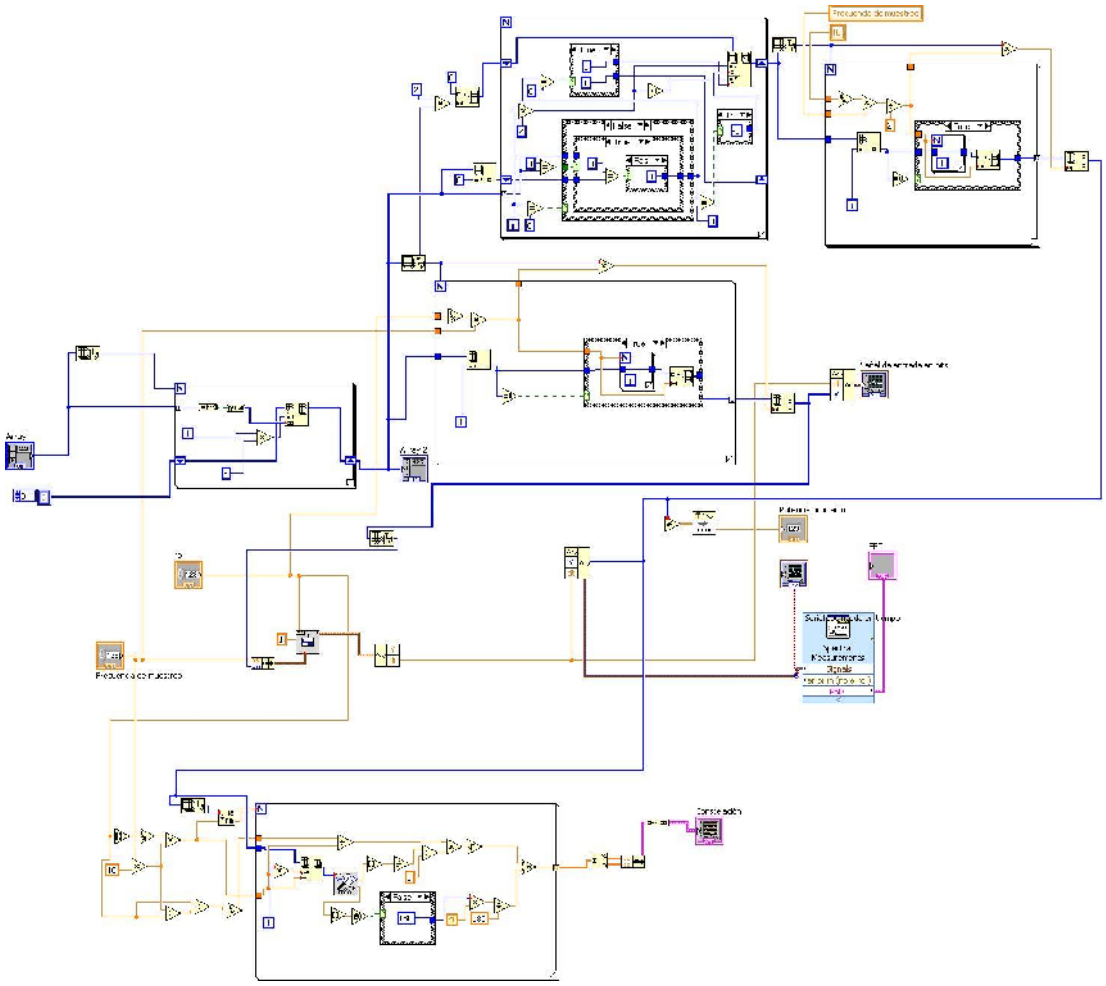


Figura 3.99: Diagrama de Bloques SubVI Código Manchester Diferencial.

3.4.9 DESARROLLO DEL PROGRAMA PRINCIPAL

Una vez que se tienen los diferentes SubVIs correspondientes a cada código de línea se procede a unirlos en el programa principal.

Primero se definen los parámetros de entrada, que serán ingresados por el usuario como son la selección del puerto para la comunicación USB, el tipo de codificación, la frecuencia y la trama de datos a ser codificados, este último parámetro permite enviar hasta 8 tramas de 8 bits cada una, estos parámetros de inicio se muestran en la Figura 3.100 tanto en el **Front Panel** como en el **Block Diagram**.

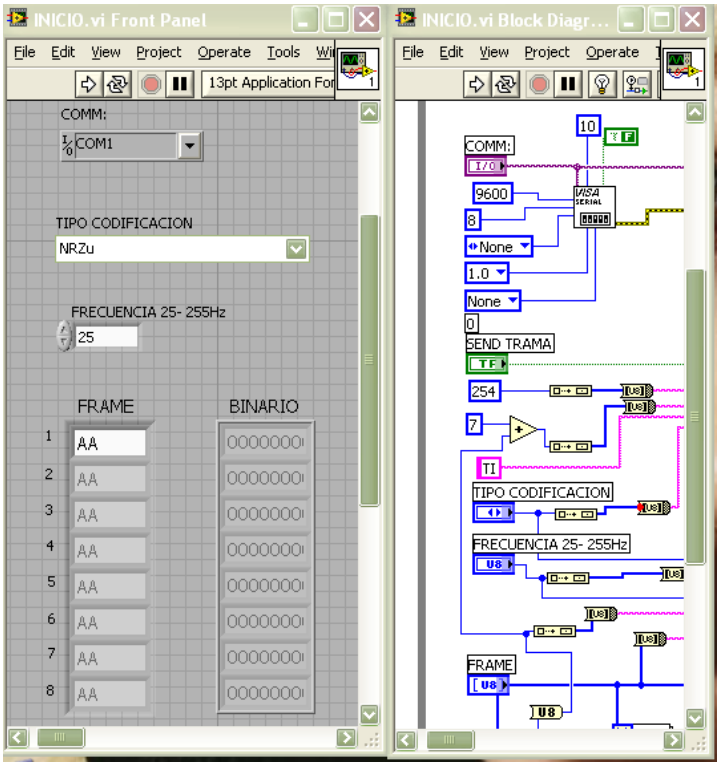


Figura 3.100: Definición de los parámetros de entrada del programa principal.

Para hacer las llamadas a los SubVIs correspondientes a los diferentes códigos de línea se crean con la ayuda de la función **Case Structure** los diferentes casos en donde se colocara el SubVI respetivo de acuerdo al código de línea de cada caso como se puede observar en la Figura 3.101.

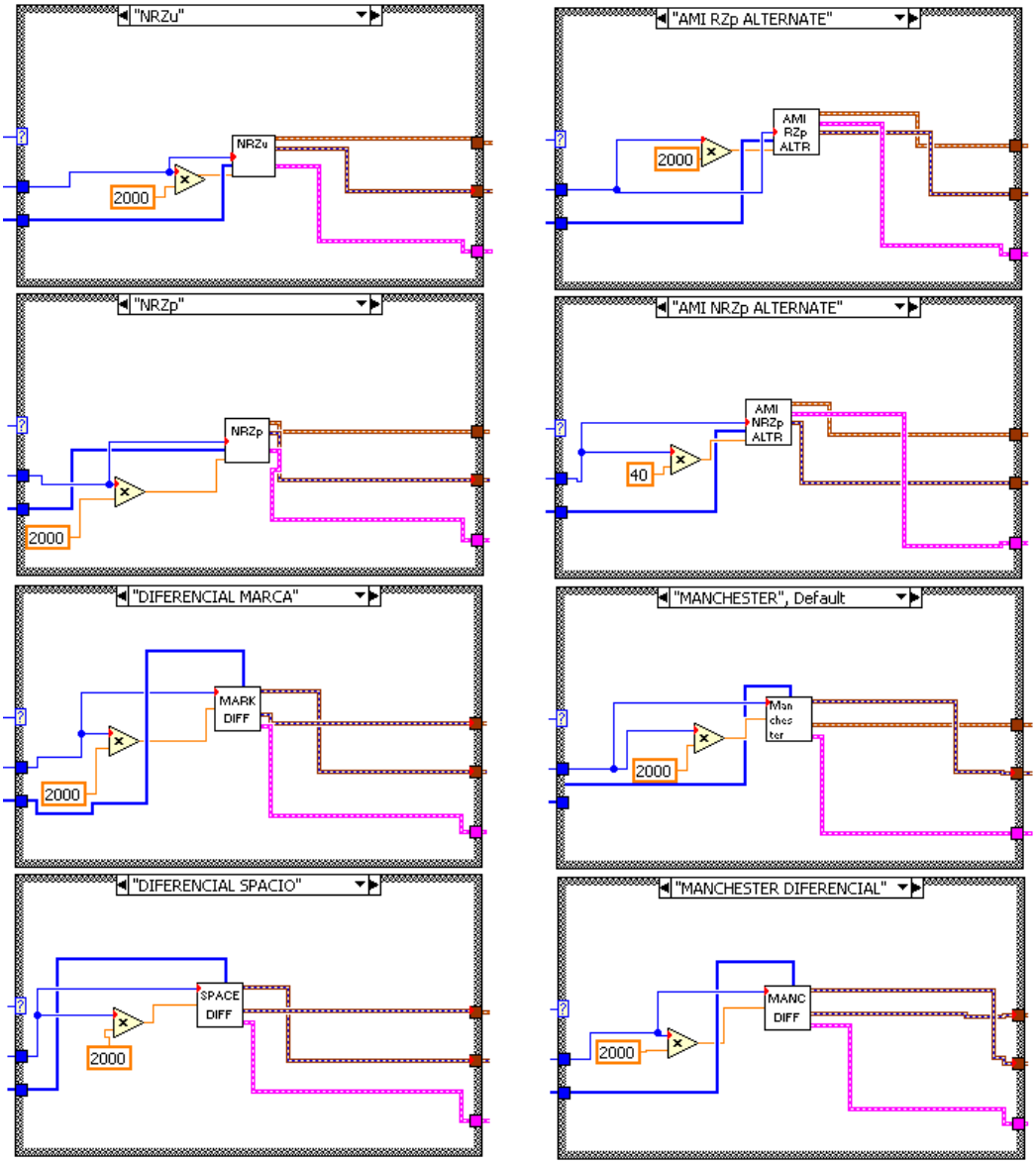


Figura 3.101: Llamada a los SubVIs de cada código de línea.

Ahora se realiza la conexión de los parámetros de entrada que son requisito de cada SubVI y las conexiones a los Indicadores en donde se visualizará la señal de entrada en bits, la señal codificada en función del tiempo y la señal codificada en función de la frecuencia, estas conexiones se observan en la Figura 3.102.

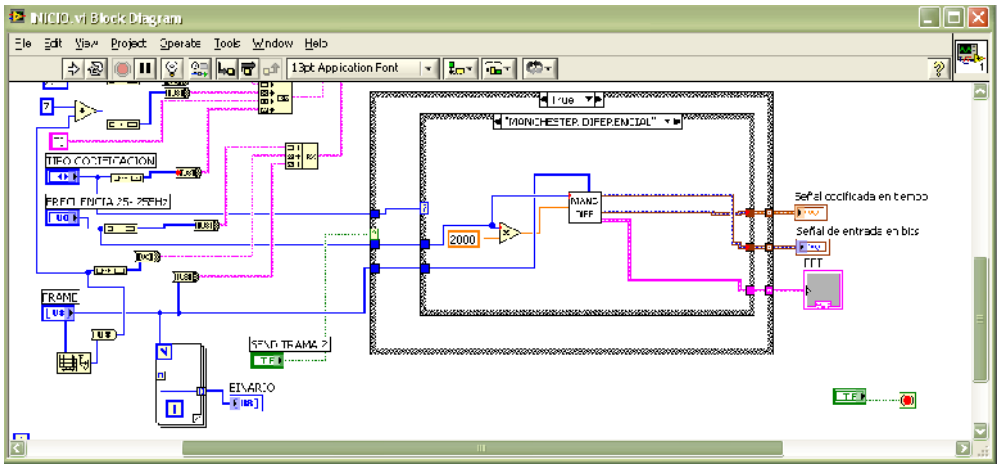


Figura 3.102: Visualización de los diferentes tipos de señales

Ahora se configura la comunicación que tendrá el software a través del puerto USB con el equipo tal como se indica en la Figura 3.103.

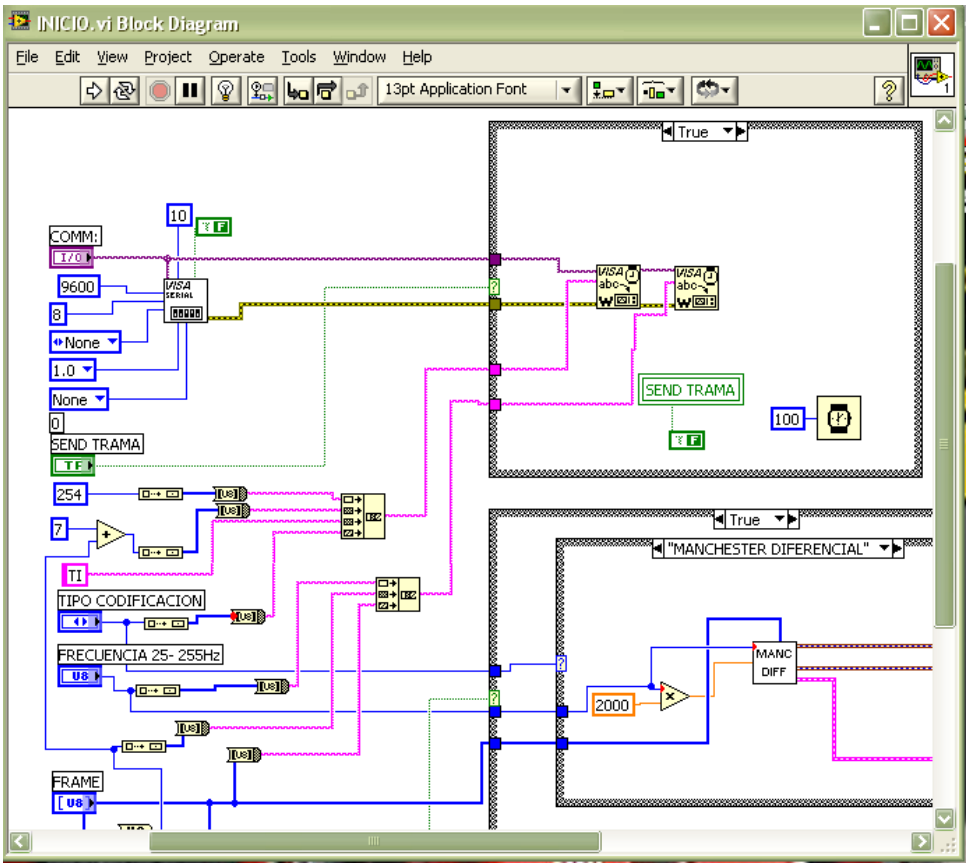


Figura 3.103: Comunicación USB.

Finalmente todas las funciones se ubican dentro de la sentencia **While Loop** para que el programa sea repetitivo, se colocan los botones para activar la Simulación, el

envió de la trama a través del puerto USB y detener el programa quedando de esta forma el Front Panel y el Block Diagram como en la Figura 3.104 y la Figura 3.105 respectivamente.

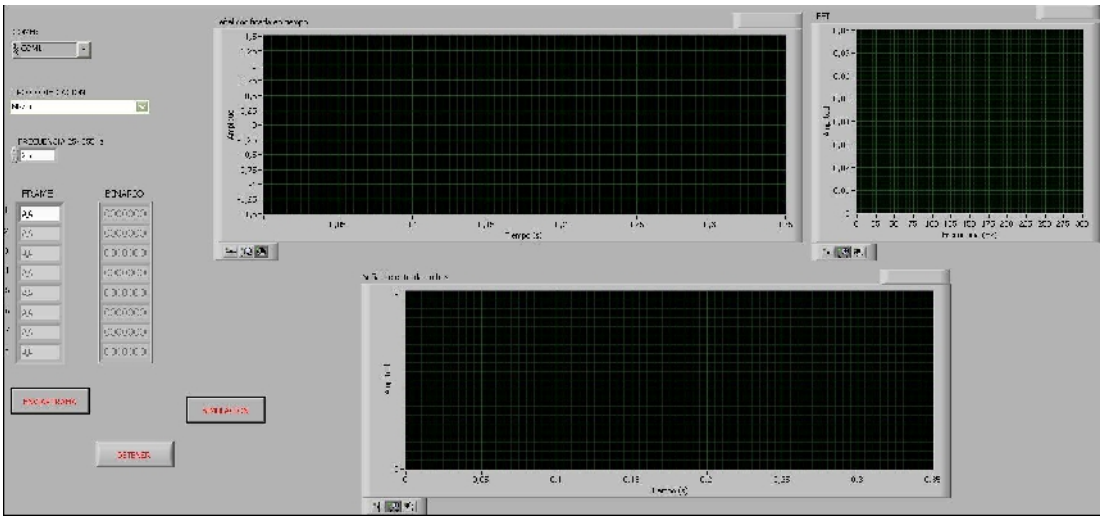


Figura 3.104: Panel Frontal del Programa Principal.

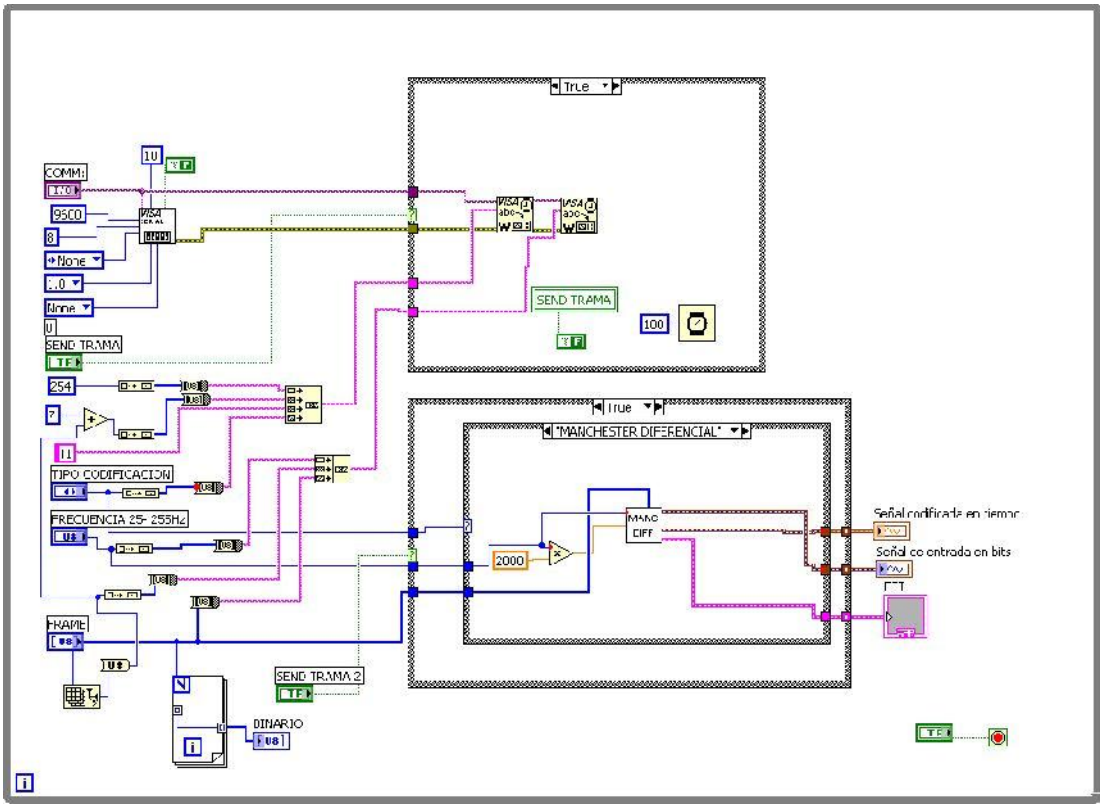


Figura 3.105: Diagrama de Bloques del Programa Principal.

3.4.10 DESARROLLO DEL PROGRAMA DEL DECODIFICADOR

Este programa se desarrolló para visualizar los datos recibidos después de realizar el proceso de decodificación en el módulo respectivo, este programa únicamente permite escoger el código de línea con el que trabajara el decodificador este programa tanto en el Front Panel como en el Block Diagram se muestra en la Figura 3.106 y la Figura 3.107 respectivamente, estos diagramas se los puede visualizar en el Anexo 6 que son obtenidos directamente desde el software LabView.

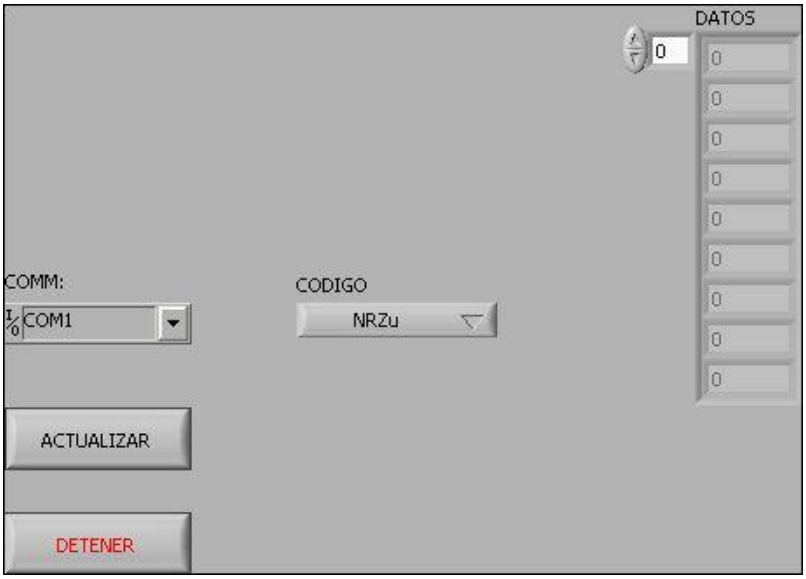


Figura 3.106: Panel Frontal Programa del Decodificador

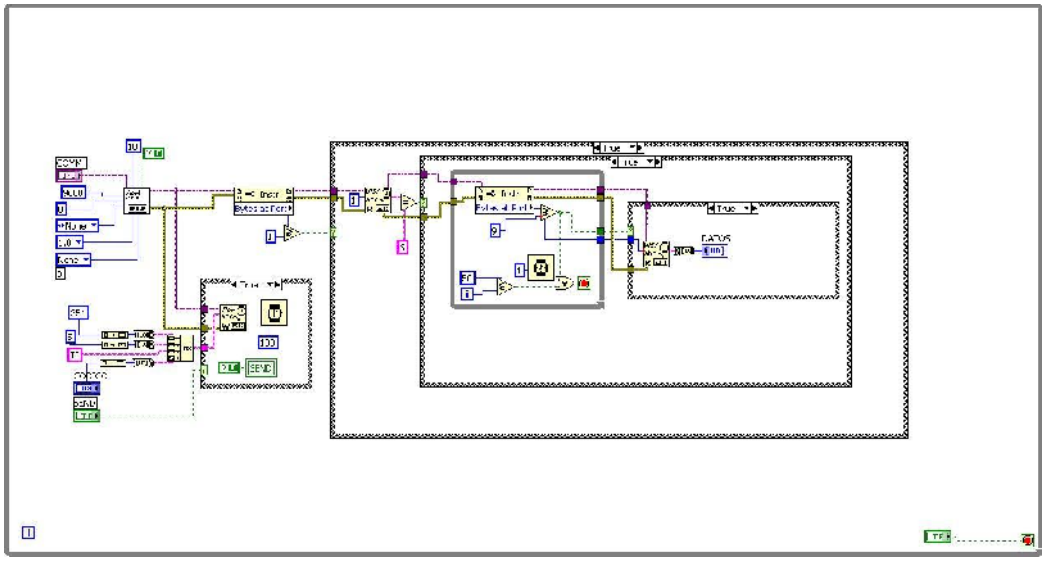


Figura 3.107: Diagrama de Bloques del Programa del Decodificador.

3.5 IMPLEMENTACION DE LOS MODULOS CODIFICADOR Y DECODIFICADOR

3.5.1 IMPLEMENTACION DEL MODULO CODIFICADOR

El módulo codificador del presente trabajo está armado siguiendo el diagrama esquemático que se presenta en la Figura 3.108, en donde se puede observar el circuito correspondiente a la fuente de energía que proporciona los niveles de voltaje positivos y negativos necesarios para la codificación de línea, además de las conexiones de los puertos del PIC18F4550 para el manejo del LCD, teclado y conexión USB.

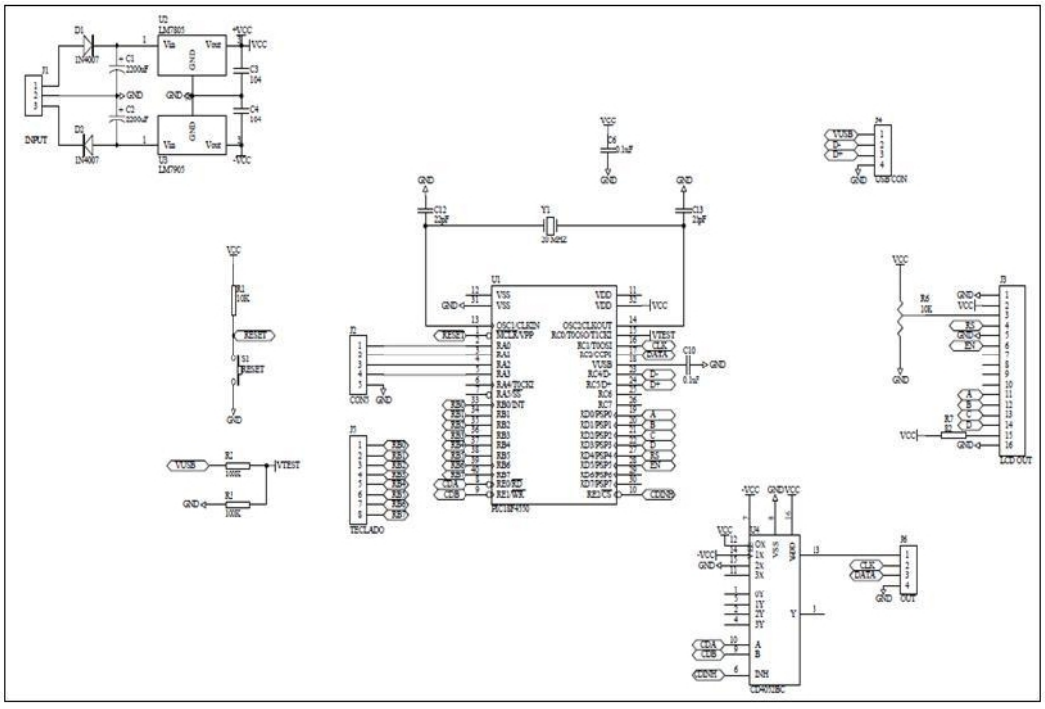


Figura 3.108: Diagrama de esquemático del módulo codificador

Del esquema anterior y con la ayuda del software Proteus y la herramienta de enrutado ARES se obtuvo el diseño del circuito impreso correspondiente al módulo codificador, como se puede observar en la Figura 3.109.

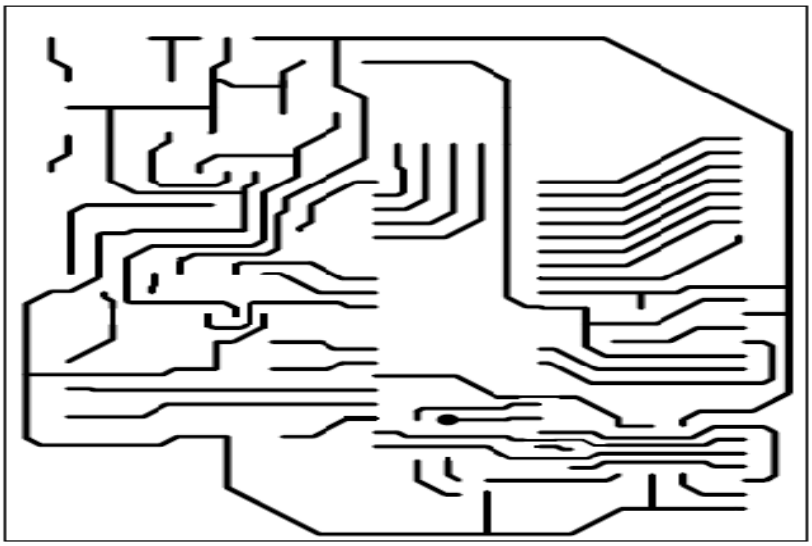


Figura 3.109: Pistas del Módulo Codificador

Adicionalmente en la Figura 3.110 se observa las posiciones de los diferentes elementos empleados para la construcción del Módulo Codificador.

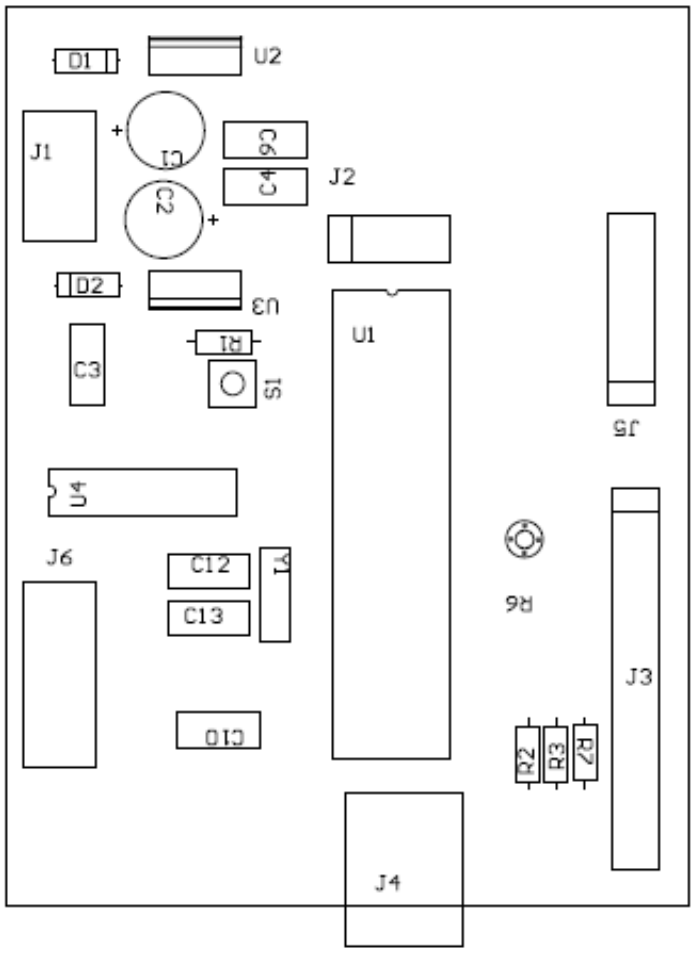


Figura 3.110: Distribución de los elementos del Módulo Codificador

En la Figura 3.111, se muestra el circuito impreso con los elementos de la fuente, la conexión USB y los terminales de salida para la conexión con el Osciloscopio, mientras que en la Figura 3.112 se muestra el circuito impreso con todos los elementos colocados y los buses para la conexión del teclado y del LCD.

Figura 3.111: Circuito Impreso Módulo Codificador

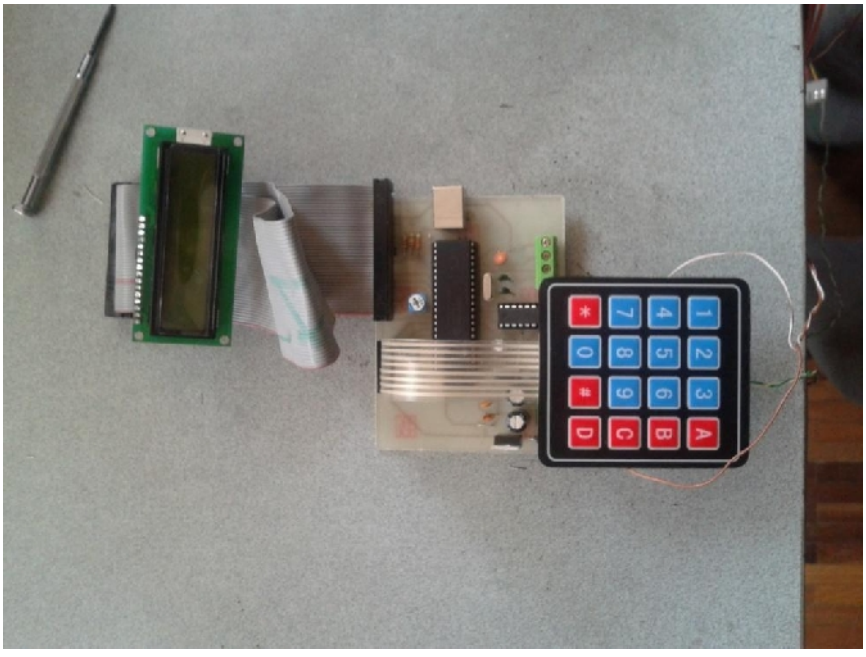


Figura 3.112: Placa con los elementos y conectores del Módulo Codificador.

Por último en la Figura 3.113, se muestra el módulo codificador con todos sus elementos y conexiones respectivas instaladas en la caja de proyectos, y en la Figura 3.114, se puede observar el módulo terminado.

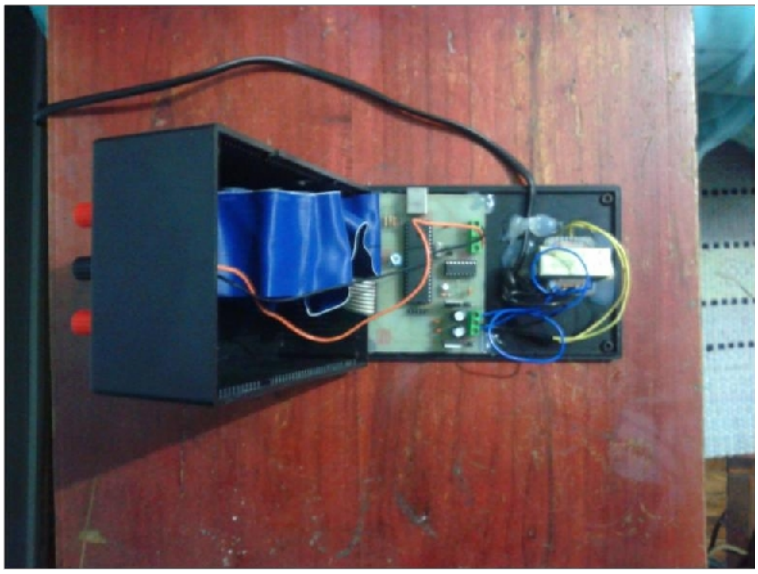


Figura 3 113: Circuito armado e instalado en la caja.

Figura 3 114: Módulo Codificador Terminado.

3.5.2 IMPLEMENTACION DEL MODULO DECODIFICADOR

El módulo decodificador al igual que el anterior se encuentra armado siguiendo el diagrama esquemático que se presenta en la Figura 3.115, en donde también se puede visualizar el circuito correspondiente a la fuente de energía, además de las conexiones de los puertos del PIC18F4550 para el manejo del LCD, los tres botones y la conexión USB.

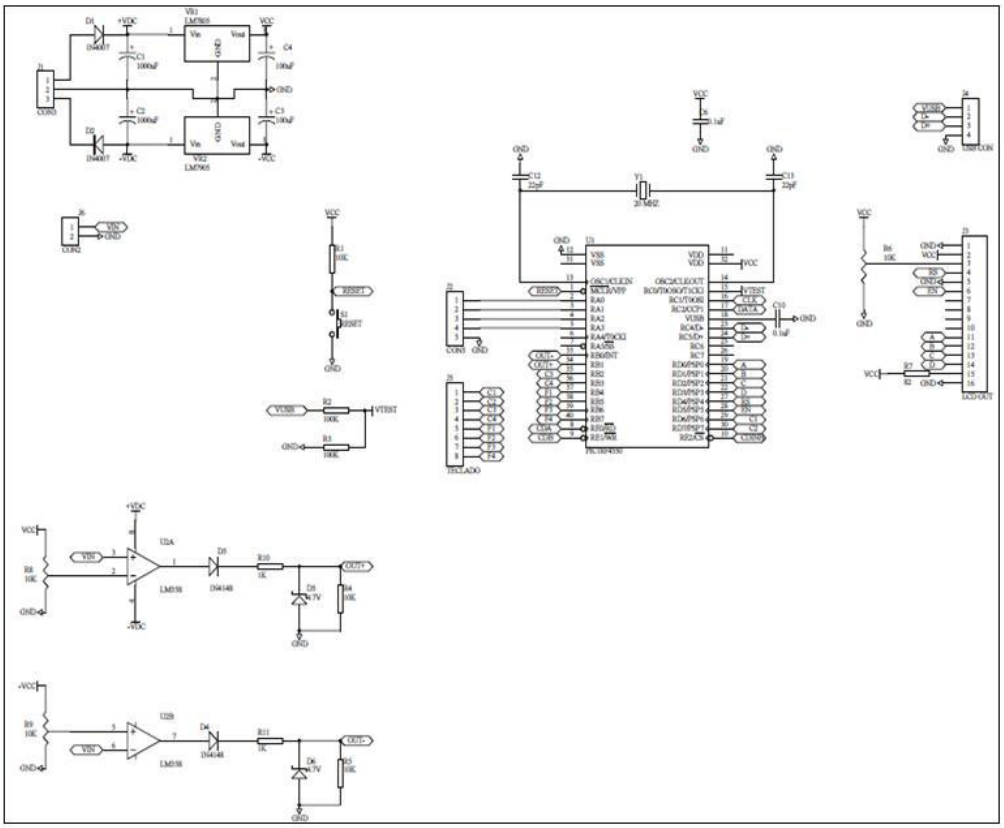


Figura 3.115: Diagrama Esquemático del módulo decodificador.

Del esquema de la Figura 3.115 y con la ayuda del software Proteus y su herramienta de enrutado ARES se obtuvo el diseño del circuito impreso correspondiente al módulo decodificador, como se puede observar en la Figura 3.116.

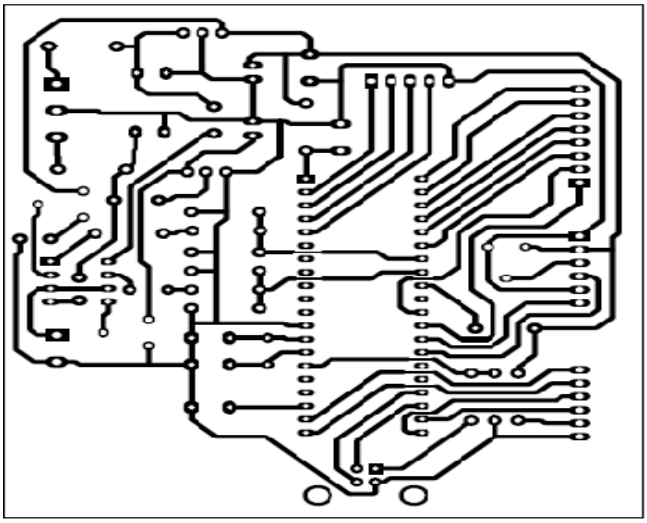


Figura 3.116: Pistas del Módulo Decodificador

Al igual que para el módulo codificador en la Figura 3.117, se puede observar las posiciones de los diferentes elementos empleados para la construcción del Módulo Decodificador.

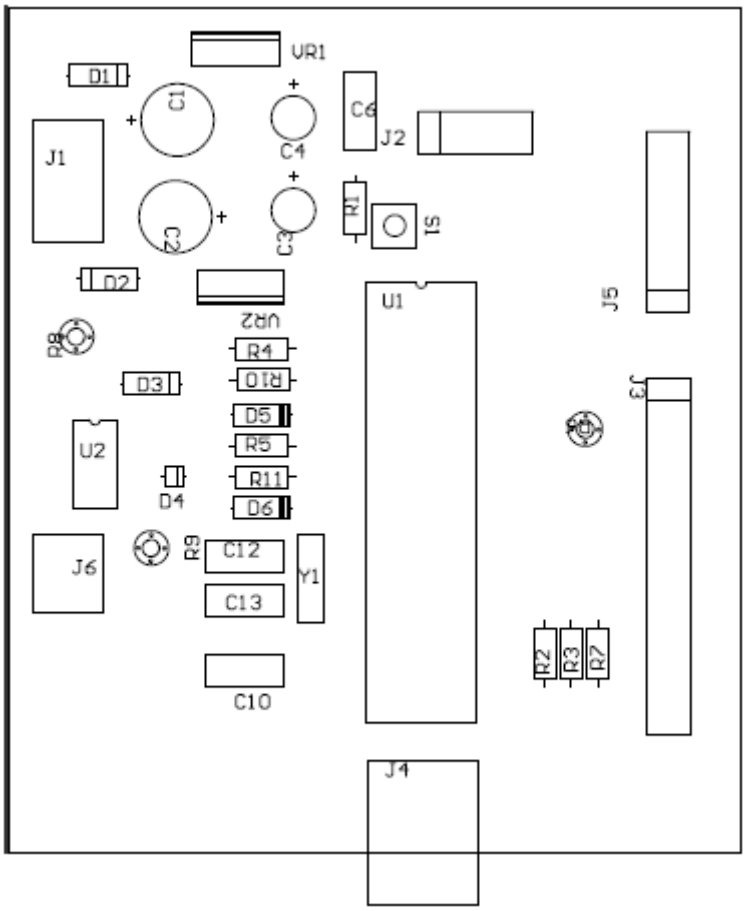


Figura 3 117: Distribución de los elementos del Módulo Decodificador

En la Figura 3.118, se muestra el circuito impreso con los elementos de la fuente, la conexión USB y los terminales de entrada.

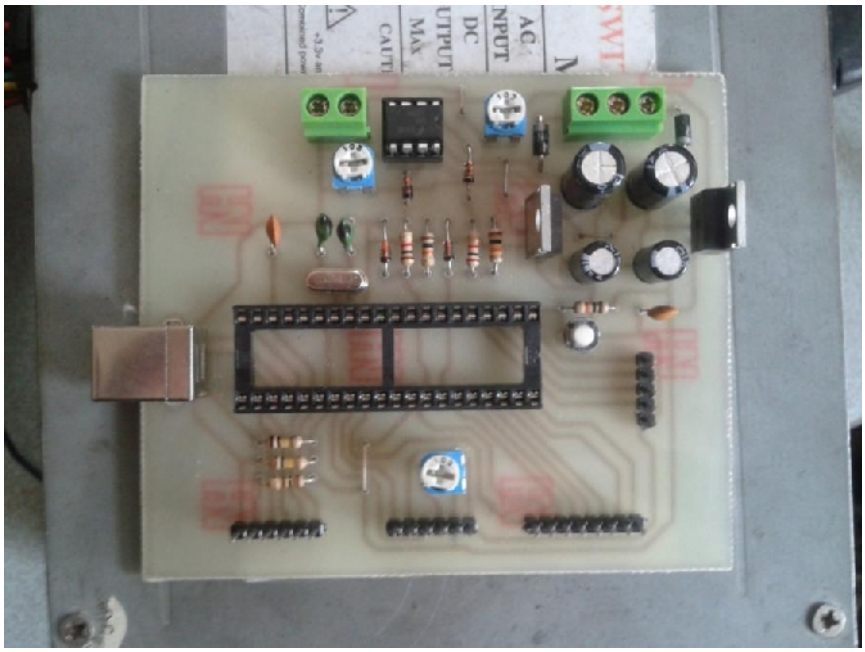


Figura 3.118: Circuito Impreso Módulo Decodificador

Por último en la Figura 3.119, se muestra el módulo codificador con todos sus elementos y conexiones respectivas instaladas en la caja de proyectos, y en la Figura 3.120, se puede observar el módulo decodificador terminado.

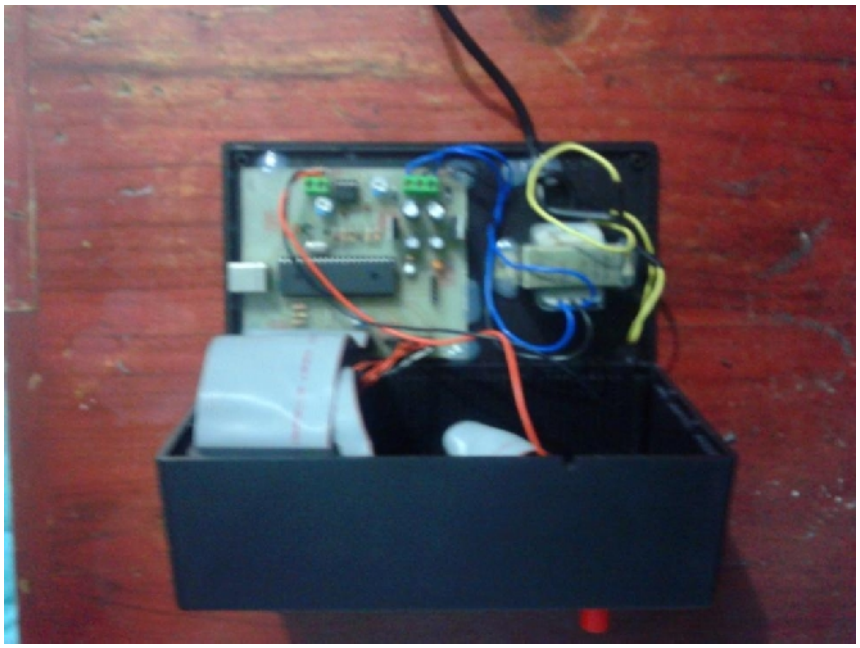


Figura 3.119: Circuito armado e instalado en la caja.

Figura 3.120: Módulo Decodificador Terminado

CAPITULO IV

Análisis de Resultados

4.1 INTRODUCCION

En este capítulo, se presentarán los resultados de las pruebas de funcionamiento que se realizaron tanto al Módulo Codificador como al Módulo Decodificador con el fin de comprobar que los procesos de Codificación y Decodificación de los datos ingresados no presenten errores.

4.2 REQUERIMIENTOS PARA EL FUNCIONAMIENTO DEL EQUIPO

Para que el módulo diseñado e implementado tenga un correcto funcionamiento es necesario que la maquina en la cual va a ser utilizado tenga las siguientes características:

- Sistema Operativo Windows XP o superior.
- Memoria RAM 2GB.
- Procesador Dual Core de 1.3 GHz.

4.3 PRUEBAS DE FUNCIONAMIENTO

Para comprobar el funcionamiento de los equipos se probó cada código de línea implementado mediante la visualización en el osciloscopio, aprovechando que estos equipos de medición del laboratorio permiten guardar el resultado en una imagen se obtuvieron los siguientes resultados:

4.3.1 PRUEBAS DE CONEXION DE LOS EQUIPOS

La primera prueba que se realizó es la conexión desde el módulo codificador al módulo decodificador para verificar que los datos que están enviando se están recibiendo sin errores, en la Figura 4.1 se puede observar esta conexión.

Figura 4.1: Funcionamiento de los equipos.

La siguiente prueba fue la realización de la conexión con la PC para verificar el funcionamiento del Software desarrollado, en la Figura 4.2 se puede observar el funcionamiento del mismo.

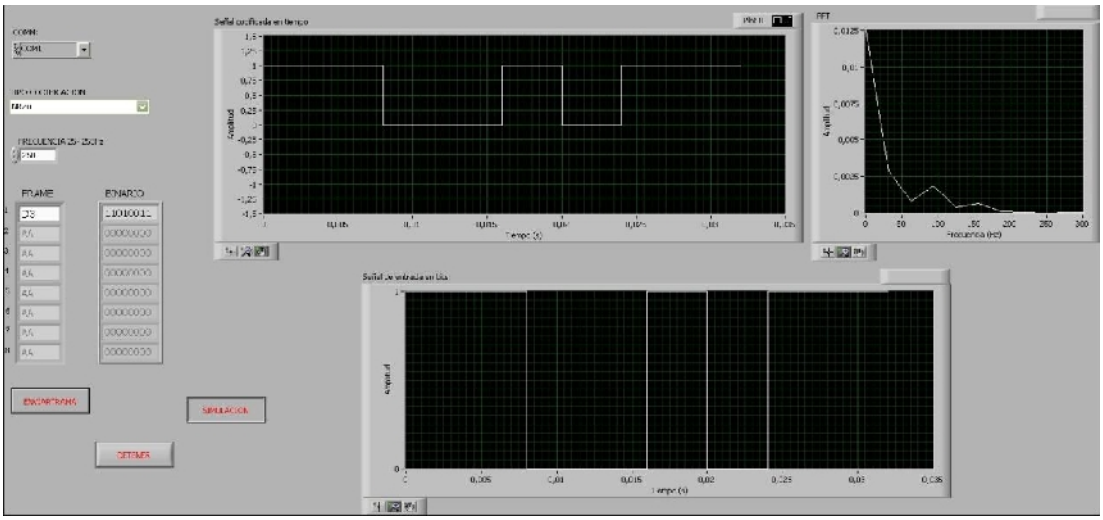


Figura 4.2: Funcionamiento del Software

La última prueba que se realizó fue la conexión al osciloscopio para verificar el envió y codificación de los datos como se muestra en la Figura 4.3.

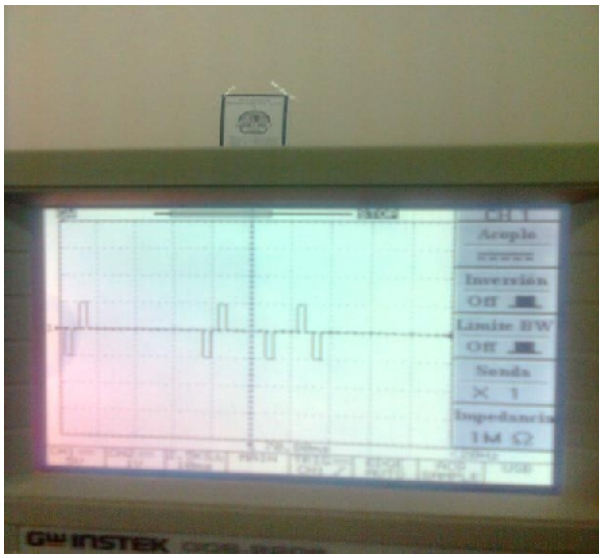


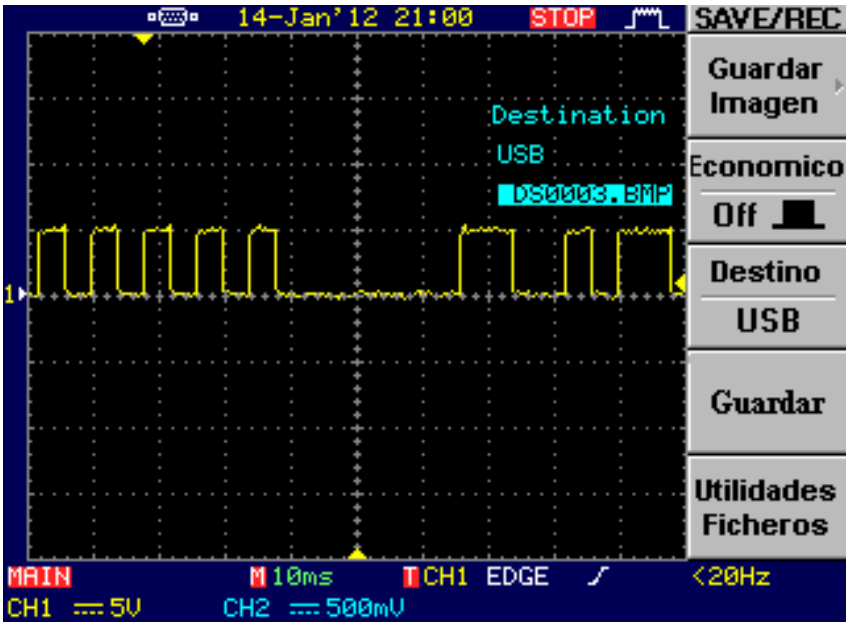
Figura 4.3: Visualización de los datos en el Osciloscopio

4.3.2 RESULTADOS OBTENIDOS

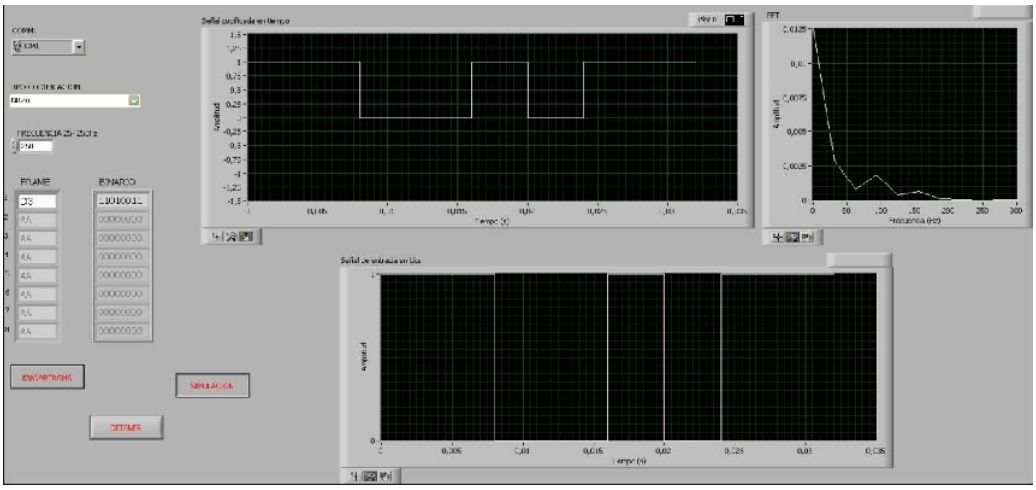
Los resultados que se obtuvieron en el Osciloscopio al ingresar el dato 11010011 o su equivalente hexadecimal D3 para cada uno de los Códigos de Línea implementados fueron:

Código de Línea NRZu.

- Resultado Osciloscopio



- Resultados Simulación en el Software



Como se puede observar en la gráfica para la codificación de línea no retorno a cero unipolar (NRZu) obtenida en el osciloscopio, esta presenta al inicio los bits de sincronización los cuales son utilizados para que el decodificador cumpla con la función de obtener los datos enviados desde el módulo codificador sin errores, acorde al código utilizado.

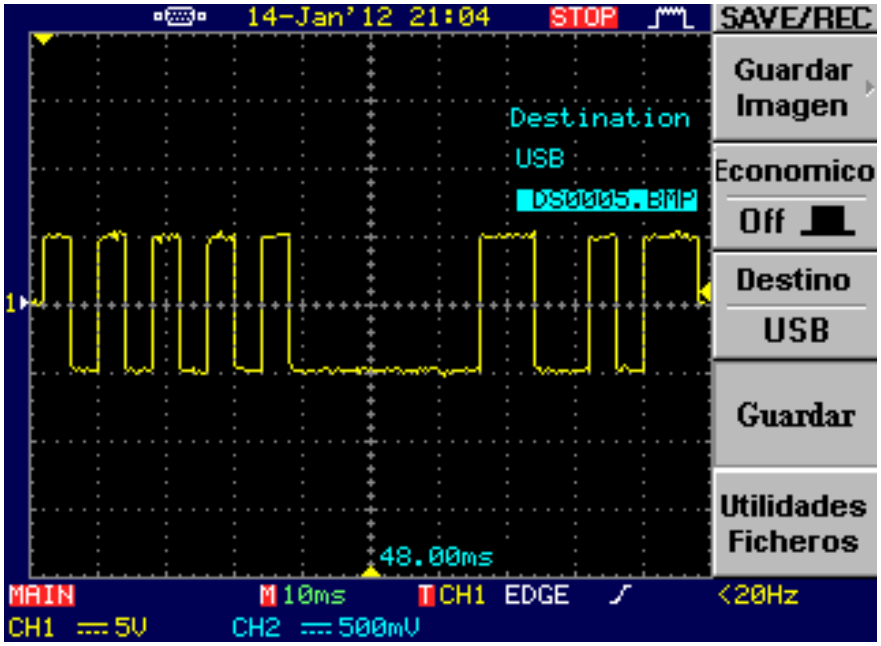
A continuación, se muestra el tiempo de espera el cual es utilizado para que el módulo codificador envié el número de datos a ser transmitidos al decodificador.

Además al comparar las dos gráficas tanto la simulada en el software como en el osciloscopio se puede observar que son muy similares y están de acuerdo al algoritmo de codificación desarrollado.

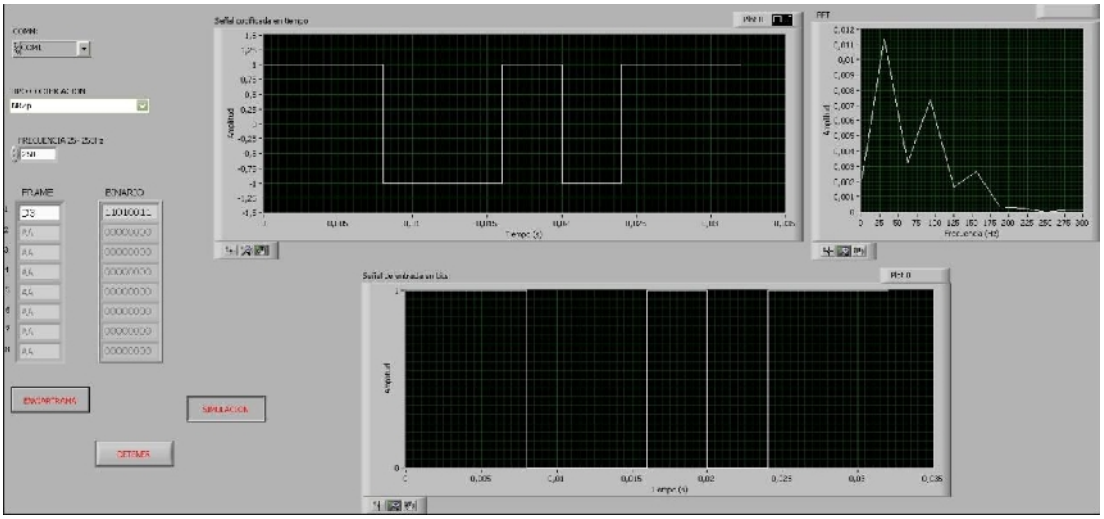
Este análisis es similar para todos los códigos de línea probados y cuyas gráficas se muestran a continuación.

Código de Línea NRZp.

- Resultado Osciloscopio

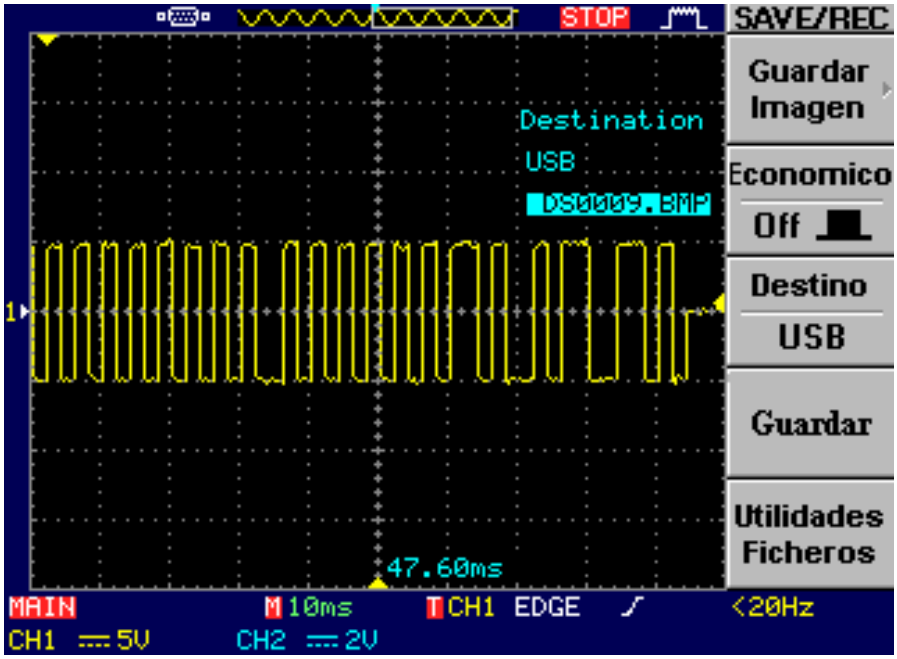


- Resultados Simulación en el Software

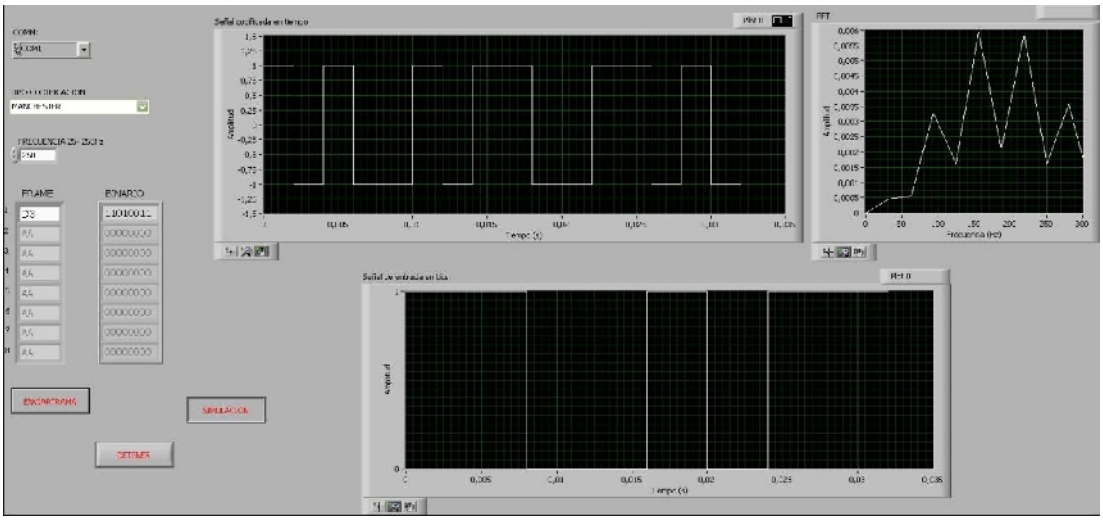


Código de Línea Manchester.

- Resultado Osciloscopio

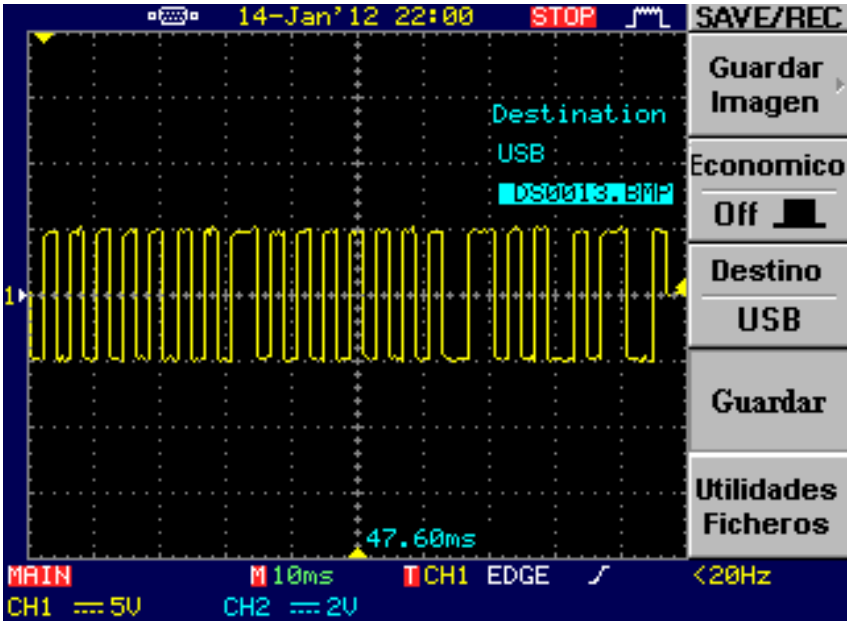


- Resultados Simulación en el Software

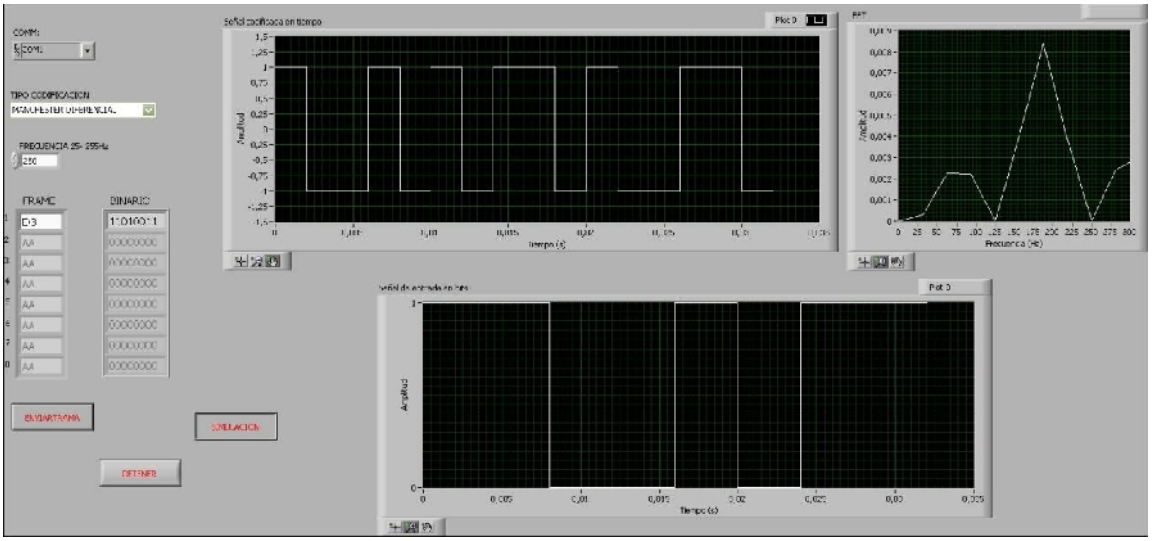


Código de Línea Manchester Diferencial.

- Resultado Osciloscopio

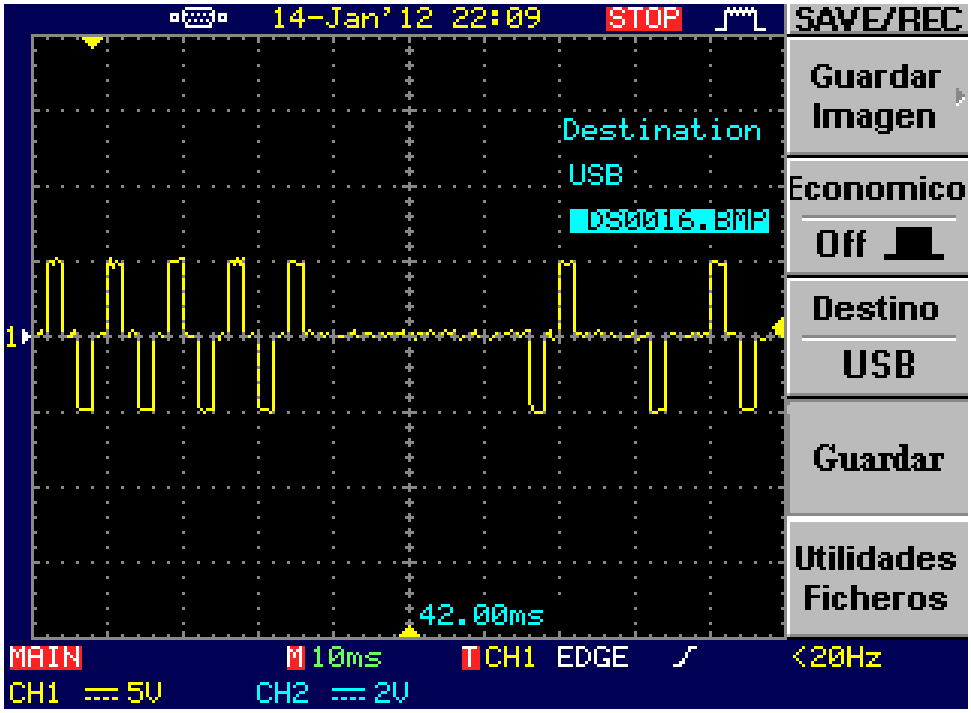


- Resultados Simulación en el Software

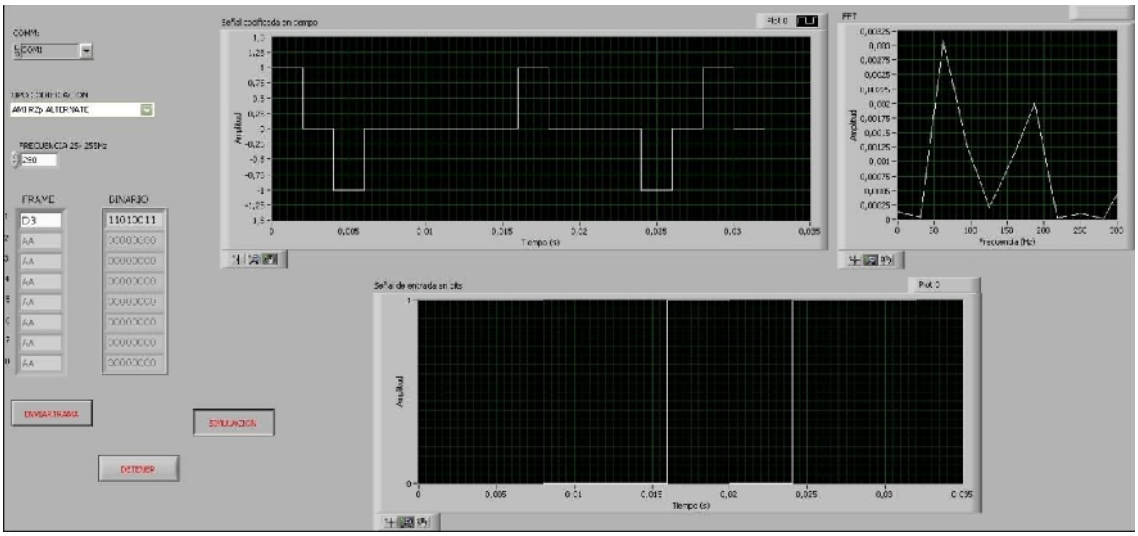


Código de Línea AMIRZp.

- Resultado Osciloscopio

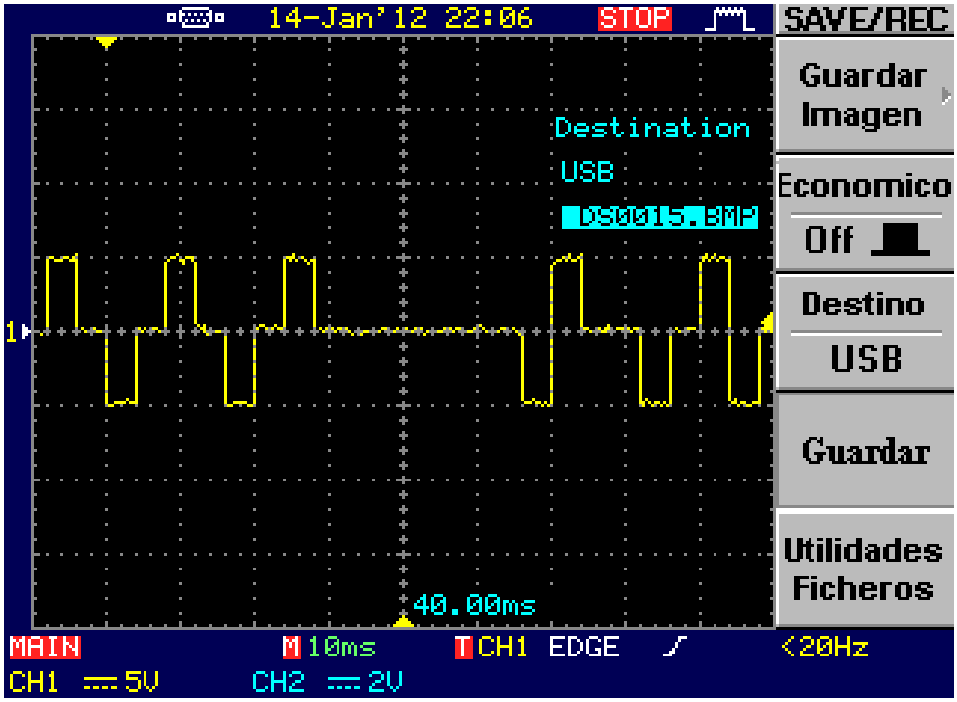


- Resultados Simulación en el Software

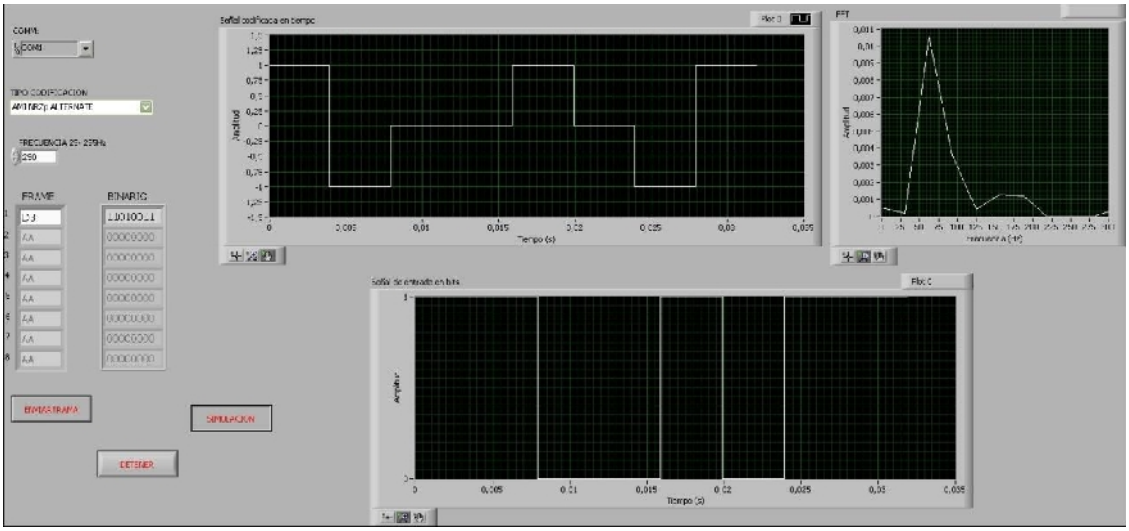


Código de Línea AMINRZp.

- Resultado Osciloscopio

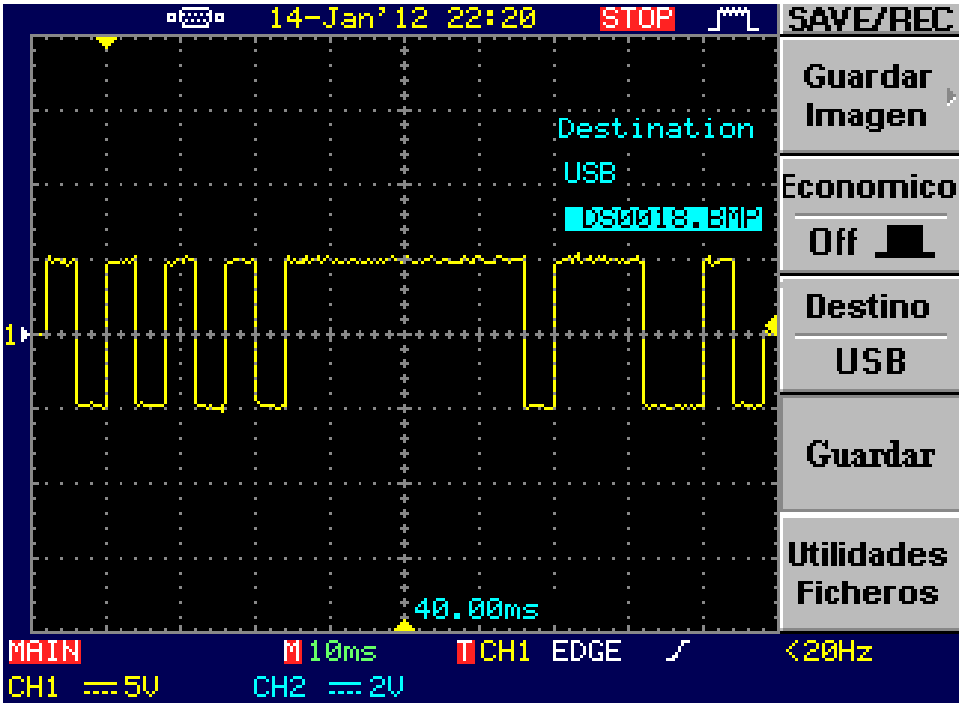


- Resultados Simulación en el Software

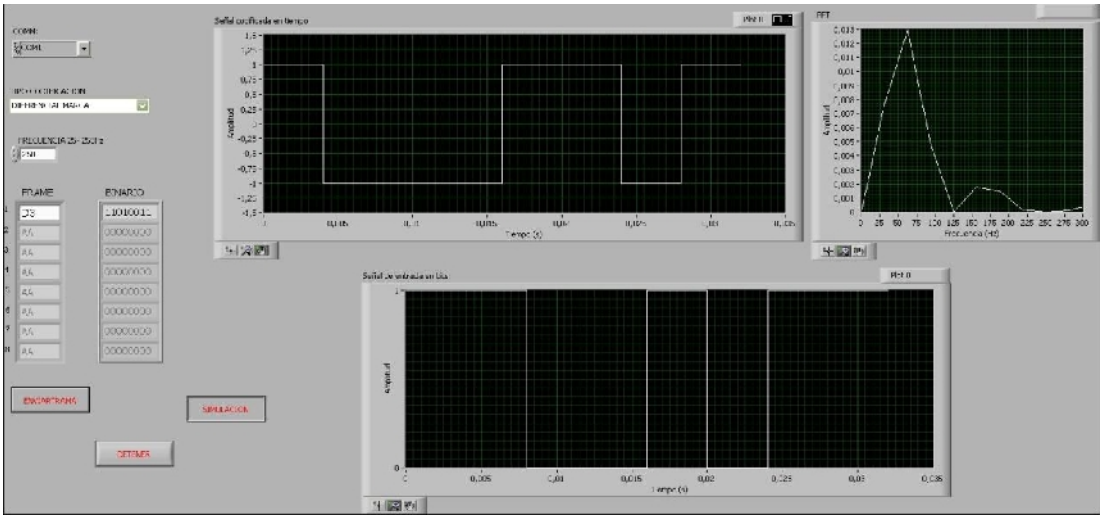


Código de Línea Diferencial tipo Marca.

- Resultado Osciloscopio

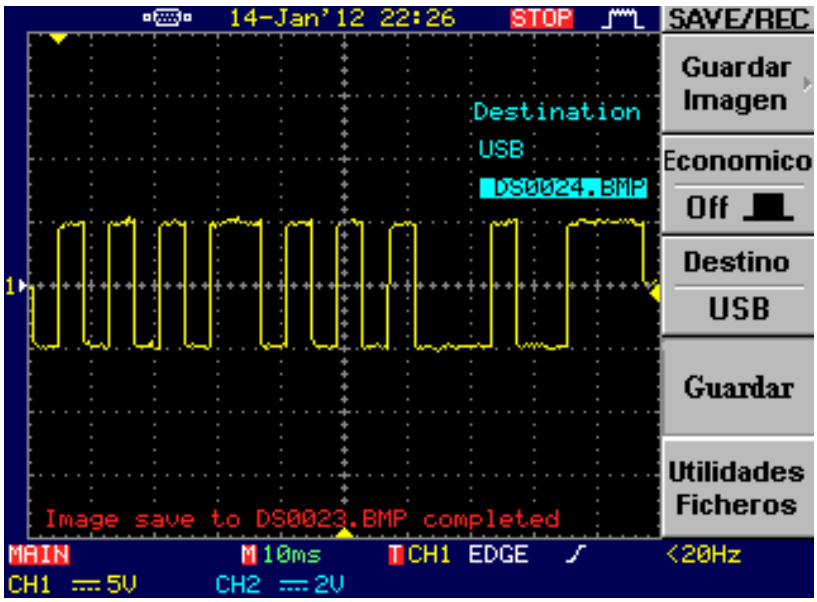


- Resultados Simulación en el Software

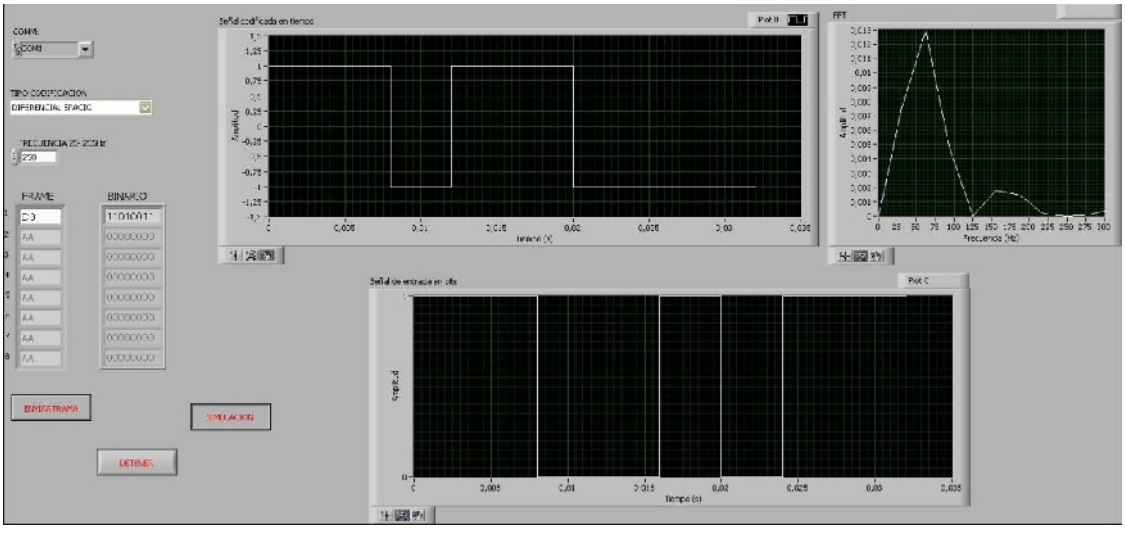


Código de Línea Diferencial tipo Espacio.

- Resultado Osciloscopio



- Resultados Simulación en el Software



CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- Para mejorar la visualización de la señal codificada en el Osciloscopio se utilizo una punta de prueba por 10 de precisión debido a que la misma posee un filtro interno el cual ayuda a eliminar señales de ruido externas al equipo.

- Para que exista una correcta comunicación entre la PC y el dispositivo Microcontrolador PIC, se utilizó en el presente proyecto el software compilador PICC debido a que este permite crear drivers que son soportados en diferentes Sistemas Operativos para lograr la comunicación USB, que con otros compiladores no es posible desarrollarlos.

- Los algoritmos de codificación de cada unos de los códigos de línea implementados, se los desarrolló en el software de LabView en el cual las diferentes funciones del tipo Array y complementadas con otras como el For Loop, Case Structure, se realizan diversas operaciones con los datos ingresados para cumplir con los algoritmos de codificación y obtener la visualización de cada uno de ellos, de esta forma se proporciona una base sólida al momento de evaluar el correcto funcionamiento del módulo.

- Para cumplir con una de las características que poseen los códigos de línea como es la autosincronización, el módulo codificador del presente proyecto envía una cadena de bits de sincronización para determinar la señal de temporización, mediante la cual se identifica el tiempo correspondiente a un bit.

- Para que el módulo decodificador muestre de forma correcta los datos recibidos, el módulo codificador del presente proyecto envía una segunda cadena de bits a continuación de la de sincronización en la cual se indica en número de bytes transmitidos.

- Los resultados obtenidos en el hardware son satisfactorios, ya que las diferentes gráficas obtenidas en el osciloscopio del módulo codificador son iguales a las presentadas mediante la interfaz gráfica desarrollada en LabView. Para este logró, el diseño del circuito del módulo Codificador el cual posee el dispositivo Multiplexor CD4053BC permite el manejo de valores de tensión positivos y negativos para que estos datos sean interpretados de forma correcta por el PIC en el proceso de codificación de línea.

RECOMENDACIONES

- Es recomendable que los estudiantes posean un buen conocimiento en el manejo del Software de LabView, para comprender con facilidad el código del programa en caso de que deseen realizar alguna modificación o implementar alguna variante por ejemplo una grafica correspondiente al diagrama del ojo o de constelación.

- Para el buen funcionamiento del Software desarrollado es aconsejable tener instalado la última versión del software VISA, de acuerdo al sistema operativo en el que se esté trabajando para que no existan problemas al realizar la comunicación USB, debido a que este programa permite el manejo de los diferentes puertos de una computadora.

- Se recomienda que la computadora en la cual va a ser utilizado el software se encuentre actualizada y cumpla con requerimientos mínimos que se indican en el Capítulo IV para evitar problemas con el software del módulo.

BIBLIOGRAFIA

1. FERREIRA, Luis F. y otros, *MICROCONTROLADORES PIC. Sistema integrado para el auto aprendizaje,* 1ra Edición Marcombo, España, 2007.

2. ANGULO, Amusátegui y otros, *MICROCONTROLADORES PIC,* 1ra Edición, Editorial MCGraw-Hill, EEUU, 2007.

3. VIZCAINO, José y otros, *LABVIEW Entorno Gráfico de programación,* 1ra Edición, Editorial MARCOMBO S.A. Mexico, 2007.

4. MÀNUEL, Antoni y otros, *INSTRUMENTACION VIRTUAL, Adquisición procesado y análisis de señales*, 1ra Edición, EDICIONES UPC, Barcelona España, 2001.

5. JOHNSON, Gary, *LABVIEW Graphical Programming,* 4ta Edición, Editorial MCGraw-Hill, EEUU, 2000.

6. FAUNDEZ Marcos, *SISTEMAS DE COMUNICACIONES*, 1ra Edición, Editorial MARCOMBO S.A., España, 2001.

7. RUBIO Juan, *COMUNICACIONES ANALOGICAS*, 1ra Edición, Editorial Ediciones UPC, Barcelona, España 1998.

8. TOMASI Wayne, *SISTEMAS DE COMUNICACIONES ELECTRONICAS,* 4ta Edición, Editorial Pearson Educación, México, 2003.

ANEXOS

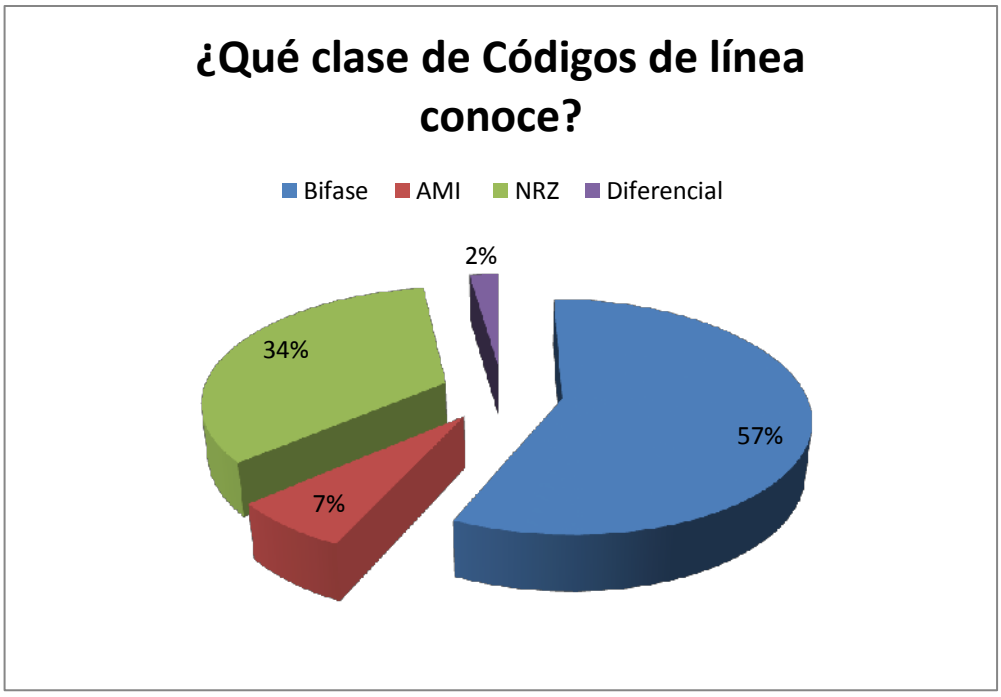
ANEXO 1

RESULTADO DEL ESTUDIO DE MERCADO

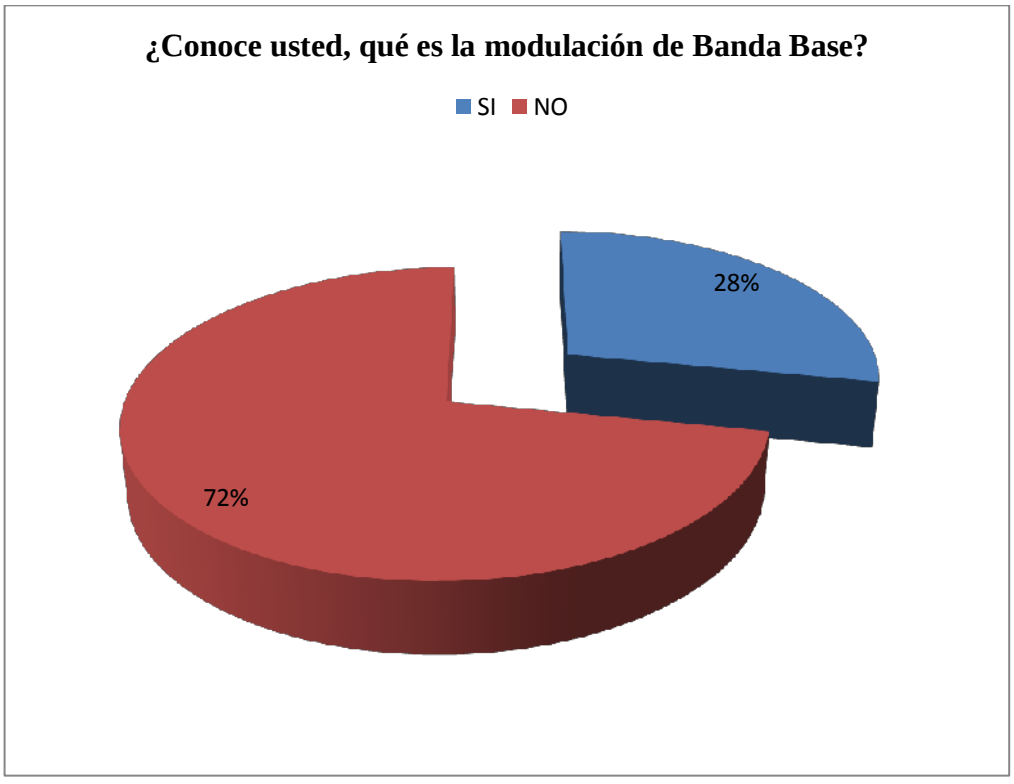
ANEXO # 1

Resultado del Estudio de Mercado

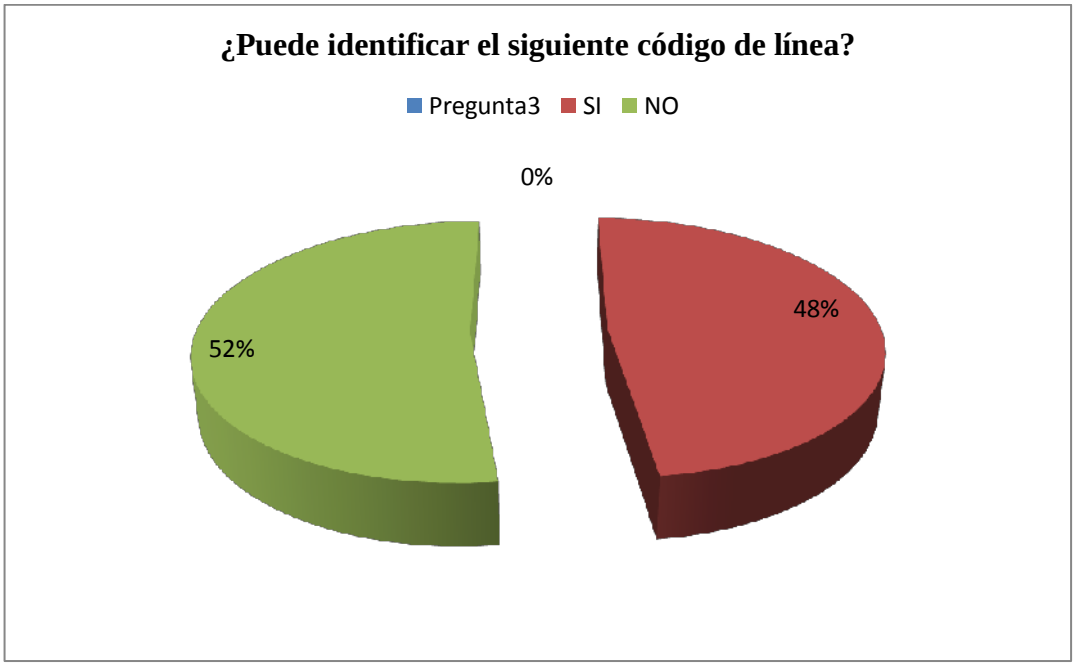
Pregunta No 1.



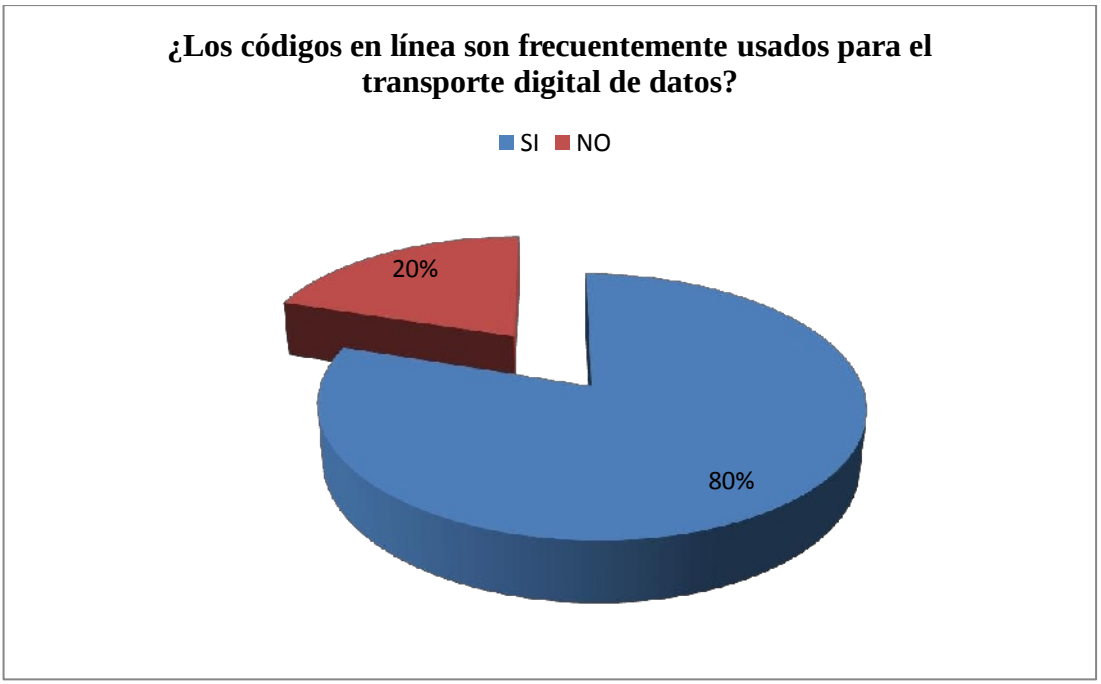
Pregunta No 2.



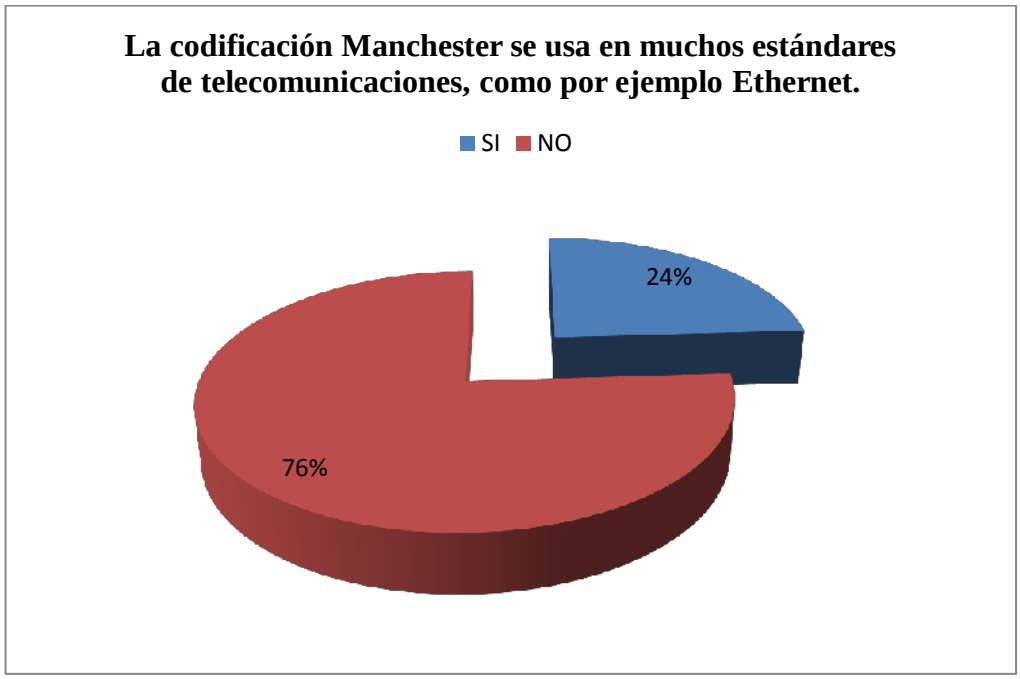
Pregunta No 3.



Pregunta No 4.



Pregunta No 5.



Pregunta No 6.

ANEXO 2

DATASHEET PIC 18F4550

ANEXO 3

DATASHEET MULTIPLEXOR

CD4053BC

ANEXO 4

CODIGO DEL CODIFICADOR

ANEXO # 4

Código del Codificador desarrollado en PIC C.

```
//set to 1 to use a PIC's internal USB Peripheral
//set to 0 to use a National USBN960x peripheral
#define __UIC_PERIF__ 1
//if using a 16bit PIC on an Explorer 16 borad, set this to 1
#define __USB_PIC_EXPLORER16__   1
   #include <18F4550.h>
   #fuses
HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,V
REGEN
   #use delay(clock=48000000)

////////////////////////////////////////////////////////////////////////////
//
// If you are using a USB connection sense pin, define it here.  If you are
// not using connection sense, comment out this line.  Without connection
// sense you will not know if the device gets disconnected.
//       (connection sense should look like this:
//                          100k
//            VBUS-----+----/\/\/\/\----- (I/O PIN ON PIC)
//                     |
//                     +----/\/\/\/\-----GND
//                          100k
//        (where VBUS is pin1 of the USB connector)
//
////////////////////////////////////////////////////////////////////////////
///only the 18F4550 development kit has this pin
#define USB_CON_SENSE_PIN PIN_C0
#include <usb_cdc.h>


//#define fila1 pin_b0;
//#define fila2 pin_b1;
//#define fila3 pin_b2;
//#define fila4 pin_b3;
#define use_portb_kbd TRUE       // habilita el teclado en el puerto b

#include <.\kbd.c>               // libreria teclado

//#define USE_LCD_RW   0         // definicion pines lcd
#define LCD_DB4   PIN_D0
#define LCD_DB5   PIN_D1
#define LCD_DB6   PIN_D2
#define LCD_DB7   PIN_D3
#define LCD_RS    PIN_D4
#define LCD_RW    PIN_D6
#define LCD_E     PIN_D5
```

```
#INCLUDE "flex_lcd.c"           // libreria lcd
#rom int
0xf00000={0x01,0xff,0x00,0x1,0xaa,0x0a,0x0a,0x0a,0x0a,0x0a,0x0a,0x0a} // graba
memoria eeprom por defecto
// variables
unsigned int16 Framefrequency;
unsigned int16 FrameTime;
unsigned int8 datainput[50],Framedata[16],DataKBD[16];
unsigned int32 temptime;

unsigned int8 FrameMode,FrameLength;
char c,length,Ftrama,idata,i,j;
UNSIGNED CHAR TEMPBIT,framecounter,mdlbit,signbit,bitback,startframe;
unsigned char datatemp,bitcounter;
char k=0;
////////////////////////////////////////////////////////////////////////////
//
// Configure the demonstration I/O
//
////////////////////////////////////////////////////////////////////////////
 //#BYTE POUT=PORTE
#define LED2 PIN_a0
#define LED3 PIN_a1
// #define POUT(X) output_e(x)
#define LED_ON(x) output_low(x)
#define LED_OFF(x) output_high(x)
#define BUTTON_PRESSED() !input(PIN_A4)
#define BUTTON_RESET() !input(PIN_c1)
#define ENPOUT PIN_E5
////////////////////////////////

#define POSITIVO 0|0b00000000
#DEFINE NEGATIVO 1|0b00000000
#DEFINE NEUTRO   2|0b00000000
void DiferencialMarca(void)
{
  if (startframe==0)
  {  startframe++;
     if (datatemp==1)
     {  output_e(positivo);
        signbit=1;
     }
     else
     {  output_e(negativo);
        signbit=0;
     }
  }
  else
  {
     if (datatemp==1)
```

```
      {
        if (signbit==0)
        {
          output_e(positivo);
        }
        else
        output_e(negativo);
      signbit++;
      if (signbit>1)
      signbit=0;
      }
    }
}
void DiferencialEspacio(void)
{
   if (startframe==0)
   {  startframe++;
      if (datatemp==1)
      {  output_e(positivo);
        signbit=1;
      }
      else
      {  output_e(negativo);
        signbit=0;
      }
   }
   else
   {
      if (datatemp==0)
      {
        if (signbit==0)
        {
          output_e(positivo);
        }
        else
        output_e(negativo);
      signbit++;
      if (signbit>1)
      signbit=0;
      }
   }
}

void AMINRZp(void)
{
      if (datatemp==1)
      { if (signbit==0)
        output_e(positivo);
       else
        output_e(negativo);
```

```
        signbit++;
        if (signbit>1)
          signbit=0;
      }
      else
        output_e(neutro);
}


void AMIRZp(void)
{
  if (mdlbit==0)
  {
    if (datatemp==1)
    { if (signbit==0)
      output_e(positivo);
     else
      output_e(negativo);
      signbit++;
      if (signbit>1)
        signbit=0;
    }
    else
      output_e(neutro);
  }
  else
    output_e(neutro);
    mdlbit++;
    if (mdlbit>1)
      mdlbit=0;

}
void NRZp(void)
{
    if (datatemp==1)
    {output_e(positivo);
    }
    else
    {output_e(negativo);
    }
}
void NRZu(void)
{
    if (datatemp==1)
    {output_e(positivo);
    }
    else
    {output_e(neutro);
    }
}
```

```
void RZp(void)
{
   if (mdlbit==0)
     {  if (datatemp==1)
           output_e(positivo);
       else
           output_e(negativo);
     }
     else
       output_e(neutro);
     mdlbit++;
     if (mdlbit>1)
       mdlbit=0;
}
void RZu(void)
{
   if (mdlbit==0)
     {  if (datatemp==1)
           output_e(positivo);
       else
           output_e(neutro);
     }
     else
       output_e(neutro);
     mdlbit++;
     if (mdlbit>1)
       mdlbit=0;
}
void Manchester(void)
{
  if (mdlbit==0)
  {  if (datatemp==1)
     {  output_e(positivo);
       signbit=1;}
     else
     {  output_e(negativo);
       signbit=0;}
  }
  else
  {  if (signbit==1)
     {  output_e(negativo);
       signbit=0;
     }
     else
     {  output_e(positivo);
       signbit=1;
     }
  }
  mdlbit++;
  if (mdlbit>1)
```

```
  mdlbit=0;
}
void ManchesterDiferencial(void)
{  if (startframe==0)
   {  startframe++;
      if (datatemp==1)
      {  output_e(positivo);
         signbit=1;}
      else
      {   output_e(negativo);
          signbit=0;}
   }
   else
   {  if (mdlbit==0)
      {  if (datatemp==0)
         {  if (signbit==0)
            {  signbit=1;
               output_e(positivo);
            }
            else
            {  signbit=0;
               output_e(negativo);
            }
         }
      }
      else
      {  if (signbit==1)
         {  output_e(negativo);
            signbit=0;
         }
         else
         {  output_e(positivo);
            signbit=1;
         }
      }
   }
      mdlbit++;
      if (mdlbit>1)
         mdlbit=0;
}


#int_timer1
void timer1interrutpt(void)
{
   clear_interrupt(int_timer1);
   set_timer1(65535-frametime);
   if (framecounter>=framelength+2)
   {
     framecounter=0;
```

```
    sETUP_TIMER_1(T1_disabled); // timer 1 apagado
    output_e(neutro);
  }
  else
  {
  datatemp=framedata[framecounter] & tempBIT;
  datatemp=datatemp>>bitcounter;
  if ((framemode<3)||(framemode==7)||(framemode==8)||(framemode==9))
     {  tempbit=tempbit<<1;
        bitcounter++;}
  else if
((Framemode==3)||(framemode==4)||(framemode==5)||(framemode==10)||(framemo
de==6))
     {if (mdlbit==0)
        {tempbit=tempbit<<1;
         bitcounter++;
        }
     }
  if (framemode==1)
     nrzp();
  else if (framemode==2)
     nrzu();
  else if (framemode==3) //xx
     rzp();
  else if (framemode==4) // xx
     rzu();
  else if (framemode==5)
     manchester();
  else if (framemode==6)
     AMIRZp();
  else if (framemode==7)
     AMINRZp();
  else if (framemode==8)
     DiferencialMarca();
  else if (framemode==9)
     DiferencialEspacio();
  else if (framemode==10)
     ManchesterDiferencial();

  if (tempbit==0)
  {  if (mdlbit==0)
     {  tempbit=1;
        framecounter++;
        bitcounter=0;
     }
  }
  }

}
```

```
//////////////////////////////////////////////////////////////////////////
//
// usb_debug_task()
//
// When called periodically, displays debugging information over serial
// to display enumeration and connection states.  Also lights LED1 based upon
// enumeration and status.
//
//////////////////////////////////////////////////////////////////////////
void usb_debug_task(void)
{
  static int8 last_connected;
  static int8 last_enumerated;
  int8 new_connected;
  int8 new_enumerated;
  static int8 last_cdc;
  int8 new_cdc;

  new_connected=usb_attached();
  new_enumerated=usb_enumerated();
  new_cdc=usb_cdc_connected();


  last_connected=new_connected;
  last_enumerated=new_enumerated;
  last_cdc=new_cdc;
}
// set timer 1
void settimer(void)
{
  temptime=3000000;
  temptime=temptime/framefrequency;
  temptime=temptime/2;
  frametime=temptime;


}
void SetDiferencialMarca(void)
{
  frametime=frametime;
  set_timer1(65535-frametime);
  framecounter=0;
  framemode=8;
  tempbit=0b00000001;
  startframe=0;
  bitcounter=0;
  mdlbit=0;
  SETUP_TIMER_1(T1_DIV_BY_8|T1_INTERNAL); // arranca timer 1
}
void SetDiferencialEspacio(void)
```

```
{
  frametime=frametime;
  set_timer1(65535-frametime);
  framecounter=0;
  framemode=9;
  tempbit=0b00000001;
  startframe=0;
  bitcounter=0;
  mdlbit=0;
  SETUP_TIMER_1(T1_DIV_BY_8|T1_INTERNAL);
}

void SetAMIRZp(void)
{
  frametime=frametime/2;
  set_timer1(65535-frametime);
  framecounter=0;
  framemode=6;
  tempbit=0b00000001;
  signbit=0;
  mdlbit=0;
  bitcounter=0;
  SETUP_TIMER_1(T1_DIV_BY_8|T1_INTERNAL);
}
void SetAMINRZp(void)
{
  frametime=frametime;
  set_timer1(65535-frametime);
  framecounter=0;
  framemode=7;
  tempbit=0b00000001;
  signbit=0;
  bitcounter=0;
  mdlbit=0;
  SETUP_TIMER_1(T1_DIV_BY_8|T1_INTERNAL);
}

// configura tiempo nrz framemode=1
void SetNRZp(void)
{
  frametime=frametime;
  set_timer1(65535-frametime);
  framecounter=0;
  framemode=1;
  tempbit=0b00000001;
  bitcounter=0;
  mdlbit=0;
  SETUP_TIMER_1(T1_DIV_BY_8|T1_INTERNAL);
}
void SetNRZu(void)
```

```
{
  frametime=frametime;
  set_timer1(65535-frametime);
  framecounter=0;
  framemode=2;
  tempbit=0b00000001;
  SETUP_TIMER_1(T1_DIV_BY_8|T1_INTERNAL);
  bitcounter=0;
  mdlbit=0;
}
void SETRZp(void)
{
  frametime=frametime/2;
  set_timer1(65535-frametime);
  framecounter=0;
  framemode=3;
  tempbit=0b00000001;
  mdlbit=0;
  bitcounter=0;
  SETUP_TIMER_1(T1_DIV_BY_8|T1_INTERNAL);
}
void SETRZu(void)
{
  frametime=frametime/2;
  set_timer1(65535-frametime);
  framecounter=0;
  framemode=4;
  tempbit=0b00000001;
  mdlbit=0;
  bitcounter=0;
  SETUP_TIMER_1(T1_DIV_BY_8|T1_INTERNAL);
}
void SETManchester(void)
{
  frametime=frametime/2;
  set_timer1(65535-frametime);
  framecounter=0;
  framemode=5;
  tempbit=0b00000001;
  mdlbit=0;
  startframe=0;
  bitcounter=0;
  SETUP_TIMER_1(T1_DIV_BY_8|T1_INTERNAL);
}
void SETManchesterDiferencial(void)
{
  frametime=frametime/2;
  set_timer1(65535-frametime);
  framecounter=0;
  framemode=10;
```

```
  tempbit=0b00000001;
  mdlbit=0;
  startframe=0;
  bitcounter=0;
  SETUP_TIMER_1(T1_DIV_BY_8|T1_INTERNAL);
}

// procesa códigos de línea

void ProcessFrame(void)
{  // lee los datos de la memoria eeporm
  framemode=read_eeprom(0);
  framefrequency=read_eeprom(1);
  framelength=read_eeprom(3);
  for (i=0;i<=7;++i)
  {  Framedata[i+2]=read_eeprom(4+i);}
  settimer();
  if (framemode==1)
  {  framedata[0]=0x55;
     framedata[1]=framelength;
     setNRZu();}
  else if(framemode==2)
  {  framedata[0]=0x55;
     framedata[1]=framelength;
     setNRZp();}
  else if(framemode==3)
  {  framedata[0]=0xff;
     framedata[1]=framelength;
     setmanchester();}
  else if(framemode==4)
  {  framedata[0]=0x00;
     framedata[1]=framelength;
     setmanchesterdiferencial();}
  else if(framemode==5)
  {  framedata[0]=0xff;
     framedata[1]=framelength;
     setamirzp();}
  else if(framemode==6)
  {  framedata[0]=0x55;
     framedata[1]=framelength;
     setaminrzp();}
  else if(framemode==7)
  {  framedata[0]=0xff;
     framedata[1]=framelength;
     setdiferencialmarca();}
  else if(framemode==8)
  {  framedata[0]=0x00;
     framedata[1]=framelength;
     setdiferencialespacio();}
```

```
}

void MainScreen(void)
{

}
// procesa dato entrada mediante puerto usb
void Processcommand(void)
{
  if ((datainput[2]=='T')&(datainput[3]=='I')) // comando trama codigo de linea
  {
    FrameMode=datainput[4]+1; // tipo codigo linea a codificar 1,2,.....
    Framefrequency=datainput[5]; // frecuencia código de línea
    FrameLength=datainput[6];
    for (i=0;i<=FrameLength;++i)
    {  FrameData[i]=datainput[i+7];
    }
                // actualizacion memoria eeprom
    write_eeprom(0,framemode);
    write_eeprom(1,framefrequency);
    write_eeprom(3,framelength);
    for (i=0;i<=framelength;++i)
    {  write_eeprom(4+i,framedata[i]);}

  }
}
void ReceiveData(void)
{  c=usb_cdc_getc(); //lectura datos puerto serial
  if ((c==0xfe)&&(Ftrama==0)) // peticion nueva trama
  {  idata=0;
    length=0;
    Ftrama=1;
  }
  if (Ftrama==1)
  {  if (idata==1)
    {  length=c;}
    if (idata>=(length-1))
    {  Ftrama=0;
      // lcd_putc(length+0x30);
       Processcommand();
    }
    datainput[idata]=c;
   }
   idata++;
}

void main(void)
{

  setup_adc_ports(NO_ANALOGS);
```

```
  // setup_adc(ADC_CLOCK_DIV_2);
   SETUP_TIMER_1(T1_DIV_BY_8|T1_INTERNAL); // configura al timer 1 como
temporizador y preescaler 8
   SETUP_TIMER_1(T1_disabled); // timer 1 apagado
   enable_interrupts(int_timer1); // habilita interrupcion del timer1
   enable_interrupts(global); // habilita interrupciones
   port_b_pullups(true);


   ftrama=0;
   idata=0;
   usb_init_cs();       // inicializa puerto usb
   lcd_init();          // inicializa lcd
   kbd_init();          // inicializa teclado
   output_e(NEUTRO);    // salida por defecto igual acero


   delay_ms(8);
   lcd_putc('\f');

   lcd_putc("TESIS CODIGOS");
   lcd_gotoxy(1,2);     // salta linea 2 columna 1 del lcd
   lcd_putc("DE LINEA");
//   delay_ms(100);
 //  framefrequency=40;
 //  framedata[0]=7;
 //  framedata[1]=3;
 //  SetAMirzp();

 //  processframe();

   while (TRUE)
   {
     usb_task();
     usb_debug_task();
     if (usb_cdc_kbhit()) // recepcion datos usb cdc puerto serial

     {
       Receivedata();
       if (c=='\n') {usb_cdc_putc('\r'); usb_cdc_putc('\n');}
       else if (c=='\r') {usb_cdc_putc('\r'); usb_cdc_putc('\n');}
       else if (c=='!') {printf(usb_cdc_putc,"HELLO");}

     }
   //0   demo:
     k=kbd_getc();     // LEE PUERTO
   //   lcd_gotoxy(1,2);
   //   if (k>0)
  //    lcd_putc(k);
```

```
   //   delay_ms(300);
  //    goto demo;
//       if (Button_Reset()==1)
//      {  //delay_ms(300);
//         processFrame();
//      }
      if(k!=0)
      {
        if (k=='A')    // envia trama
        {
          delay_ms(300);
          processframe();
        }
        if (k=='*')    // ingreso trama maximo 8 caracteres formato hex
        {  framedata[0]=0xaa;
          i=0;
          j=0;
          lcd_putc("\f");
          lcd_putc("Ingrese # Datos");
          lcd_gotoxy(1,2);
          do
          {

            k=kbd_getc();
            if ((k<'9')&&(k>'0'))
            {
              framelength=k-0x30;
              framedata[1]=framelength;
              lcd_putc(k);
              lcd_gotoxy(1,2);
            }
          } while(k!='D');

          do
          {
          lcd_putc('\f');
          lcd_putc("Ingrese Datos");
          lcd_gotoxy(15,2);
          lcd_putc(framelength-j+0x30);
          lcd_gotoxy(1,2);

          do
          {  k=kbd_getc();
            if ((k<='1')&&(k>='0'))
            {  dataKBD[i]=k-0x30;
              lcd_putc(k);
              i++;
              if (i>7)
              {  i=0;
               //  lcd_putc('\f');
```

```
            //  lcd_putc("Ingrese Datos");
            //  lcd_gotoxy(15,2);
            //  lcd_putc(framelength-j+0x30);
              delay_ms(8);
              lcd_gotoxy(1,2);}

          }
        } while (k!='D');

framedata[j+2]=datakbd[0]*128+datakbd[1]*64+datakbd[2]*32+datakbd[3]*16+dat
akbd[4]*8+datakbd[5]*4+datakbd[6]*2+datakbd[7];
          j++;
          k=0;
          if (j>=framelength)
          break;
        } while (K!='D');

        lcd_putc('\f');
        lcd_putc("Ingrese Codigo");
        do
        {
          k=kbd_getc();
          if ((k<'9')&&(k>'0'))
          {  framemode=k-0x30;
            lcd_gotoxy(1,2);
            lcd_putc(k);}
        }  while (k!='D');
        i=0;
        datakbd[0]=0;
        datakbd[1]=0;
        datakbd[2]=0;
        lcd_putc('\f');
        lcd_putc("Ing. FREQ 25-255");
        lcd_gotoxy(1,2);
        do
        {
          k=kbd_getc();
          if ((k<='9')&&(k>='0'))
          {  datakbd[i]=k-0x30;
            lcd_putc(k);
            i++;
            if (i>2)
            {  i=0;
              lcd_gotoxy(1,2);
            }
          }
        }  while (k!='D');
        framefrequency=datakbd[0]*100+datakbd[1]*10+datakbd[2];
        if (framefrequency<25)
          framefrequency=25;
```

```
            if (framefrequency>255)
              framefrequency=255;
              // actualizacion memoria eeprom
             write_eeprom(0,framemode);
             write_eeprom(1,framefrequency);
             write_eeprom(3,framelength);
             for (i=0;i<=7;++i)
              {
                write_eeprom(4+i,framedata[i+2]);
              }
              lcd_putc('\f');
         }
      }

   }
}
```

ANEXO 5

CODIGO DEL DECODIFICADOR

ANEXO # 5

Código del Decodificador desarrollado en PIC C.

```
//set to 1 to use a PIC's internal USB Peripheral
//set to 0 to use a National USBN960x peripheral
#define __UIC_PERIF__ 1
//if using a 16bit PIC on an Explorer 16 borad, set this to 1
#define __USB_PIC_EXPLORER16__   1
   #include <18F4550.h>
   #fuses
HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,V
REGEN
   #use delay(clock=48000000)



////////////////////////////////////////////////////////////////////////////
//
// If you are using a USB connection sense pin, define it here.  If you are
// not using connection sense, comment out this line.  Without connection
// sense you will not know if the device gets disconnected.
//       (connection sense should look like this:
//                          100k
//           VBUS-----+----/\/\/\/\----- (I/O PIN ON PIC)
//                    |
//                    +----/\/\/\/\-----GND
//                          100k
//       (where VBUS is pin1 of the USB connector)
//
////////////////////////////////////////////////////////////////////////////
///only the 18F4550 development kit has this pin
#define USB_CON_SENSE_PIN PIN_C0
#include <usb_cdc.h>

//#define USE_LCD_RW   0        // definicion pines lcd
#define LCD_DB4   PIN_D0
#define LCD_DB5   PIN_D1
#define LCD_DB6   PIN_D2
#define LCD_DB7   PIN_D3
#define LCD_RS    PIN_D4
#define LCD_RW    PIN_D6
#define LCD_E     PIN_D5
#INCLUDE "flex_lcd.c"           // libreria lcd
// variables
unsigned int32 FrameTotal;
unsigned int16 Framefrequency, SampleTime,FrameStatus;
unsigned int16 FrameTime,SetFrequency[8];
unsigned int8 datainput[30],Framedata[16],DataRx,FrameEnd;
unsigned int32 temptime;
```

```
unsigned int8 FrameMode,FrameLength;
char c,length,Ftrama,idata,i,j;
UNSIGNED CHAR
TEMPBIT,DATATEMP,framecounter,mdlbit,databack,databit,framesyncr;
char k=0,z,ftimer2delay;
unsigned int8
changes,changestemp,StartBit,FrameBit,FrameBitCounter,Temp1,Temp2;
////////////////////////////////////////////////////////////////////////
//
// Configure the demonstration I/O
//
////////////////////////////////////////////////////////////////////////
#BYTE data_in=getenv("SFR:PORTB")
//#define LED2 PIN_B4
//#define LED3 PIN_B5
#define LED_ON(x) output_low(x)
#define LED_OFF(x) output_high(x)
#define BUTTON_FRAME() !input(PIN_B0)
#define BUTTON_RESET() !input(PIN_B1)
#DEFINE BUTTON_ENT() !input(PIN_B6)
#define POSITIVO 0b00100000
#DEFINE NEGATIVO 0b00010000
#DEFINE NEUTRO   0b00000000
void NRZUDecoder(void)
{
    if (changes==positivo)
    {  Framebit=0b00000001;
       framebit=framebit<<framebitcounter;
       framebitcounter++;
    }
    else if (changes==neutro)
    {  framebit=0b00000000;
       framebit=framebit<<framebitcounter;
       framebitcounter++;
    }
    datarx=framebit|datarx;
}
void NRZPDecoder(void)
{
    if (changes==positivo)
    {  Framebit=0b00000001;
       framebit=framebit<<framebitcounter;
       framebitcounter++;
    }
    else if (changes==NEGATIVO)
    {  framebit=0b00000000;
       framebit=framebit<<framebitcounter;
       framebitcounter++;
    }
```

```
  datarx=framebit|datarx;
}
void DiferencialMarcaDecoder(void)
{   if (changes==positivo)
     {  databit=1;}
     if (changes==negativo)
     {  databit=0;}
  if ((framecounter==0)&&(framebitcounter==0)) // bit inicial
  {  if (databit==1)
     {  framebit=0b00000001;
        Framebit=frameBit<<FramebitCounter;
        framebitcounter++;}
     else if (databit==0)
     {  framebit=0b00000000;
        Framebit=frameBit<<FramebitCounter;
        framebitcounter++;}
  }
  else
  {   if (databack==databit)
     {  Framebit=0b00000000;
     Framebit=frameBit<<FramebitCounter;
     framebitcounter++;}
  else
     {  FrameBit=0b00000001;
     Framebit=frameBit<<FramebitCounter;
     FrameBitcounter++;}
  }
  databack=databit;
  datarx=framebit|datarx;
}
void DiferencialEspacioDecoder(void)
{  if (changes==positivo)
     {  databit=1;}
  if (changes==negativo)
     {  databit=0;}
  if ((framecounter==0)&&(framebitcounter==0)) // bit inicial
  {  if (databit==1)
     {  framebit=0b00000001;
        Framebit=frameBit<<FramebitCounter;
        framebitcounter++;}
     else if (databit==0)
     {  framebit=0b00000000;
        Framebit=frameBit<<FramebitCounter;
        framebitcounter++;}
  }
  else
  {  if (databack==databit)
     {  Framebit=0b000000001;
        Framebit=frameBit<<FramebitCounter;
```

```
        framebitcounter++;}
    else
      {  FrameBit=0b00000000;
        Framebit=frameBit<<FramebitCounter;
        FrameBitcounter++;}
    }
    databack=databit;
    datarx=framebit|datarx;
}
void ManchesterDecoder(void)
{ if (mdlbit==0)
    {  if (changes==positivo)
      {  FrameBit=0b00000001;
        Framebit=frameBit<<FramebitCounter;
        FrameBitcounter++;
      }
      else if (changes==negativo)
      {  FrameBit=0b00000000;
        Framebit=frameBit<<FramebitCounter;
        FrameBitcounter++;
      }
      datarx=framebit|datarx;
    }

}
void ManchesterDiferencialDecoder(void)
{
    if (changes==positivo)
    {  databit=1;}
    if (changes==negativo)
    {  databit=0;}
    if (mdlbit==1)
    {  databack=databit;}
    else if (mdlbit==0)
    {  if ((framecounter==0)&&(framebitcounter==0)) // bit inicial
      {  if (databit==1)
        {  framebit=0b00000001;
          Framebit=frameBit<<FramebitCounter;
          framebitcounter++;}
        else if (databit==0)
        {  framebit=0b00000000;
          Framebit=frameBit<<FramebitCounter;
          framebitcounter++;}
      }
      else
      { if (databit==databack)
        {   FrameBit=0b00000001;
          Framebit=frameBit<<FramebitCounter;
          FrameBitcounter++;}
        else if (databit!=databack)
```

```
      {  FrameBit=0b00000000;
         Framebit=frameBit<<FramebitCounter;
         FrameBitcounter++;}
    }
    datarx=framebit|datarx;
  }

}

void AMIRZPDecoder(void)
{
  if (mdlbit==0)
  {  if ((changes==negativo)||(changes==positivo))
    {
      FrameBit=0b00000001;
      Framebit=frameBit<<FramebitCounter;
      FrameBitcounter++;
    }
    else if (changes==neutro)
    {
      FrameBit=0b00000000;
      Framebit=frameBit<<FramebitCounter;
      FrameBitcounter++;
    }
    datarx=framebit|datarx;
  }

}
void AMINRZPDecoder(void)
{   if ((changes==negativo)||(changes==positivo))
    {
      FrameBit=0b00000001;
      Framebit=frameBit<<FramebitCounter;
      FrameBitcounter++;
    }
    else if (changes==neutro)
    {
      FrameBit=0b00000000;
      Framebit=frameBit<<FramebitCounter;
      FrameBitcounter++;
    }
  datarx=framebit|datarx;
}

// interrupcion cambio portb rb4-rb7
#INT_RB
void rb_isr(void)
{
  clear_interrupt(int_rb);
  changesTemp= input_b() & 0b00110000;
```

```
  if (FrameEnd==0)
  {  if (ftimer2delay==0)
     { ftimer2delay=1;
       setup_timer_2(T2_DIV_BY_4,150,1);
       if (startbit==1)
       {  sampleTime=get_timer1();
          if (FrameSyncr==1)
             set_timer1(65535-Frametime);
          else
          set_timer1(0);
       }
       if (startbit==0)
       {  SETUP_TIMER_1(T1_DIV_BY_8|T1_INTERNAL); // set timer 1
          startbit=1;}
     }
  }
}
// interrupcion del timer 0
#int_timer2
void timer2interrutpt(void)
{  clear_interrupt(int_timer2);
   setup_timer_2(T2_disabled,150,1);
   set_timer2(0);
   ftimer2delay=0;
   changes= input_b() & 0b00110000;
   if (frameBit==0)
    { FrameBit=0b00000001;}
    if (framemode==1)
    { NRZUDecoder();
      mdlbit=0;}
    else if (FrameMode==2)
    { NRZPDecoder();
      mdlbit=0;}
    else if (FrameMode==3)
    { ManchesterDecoder();
      mdlbit++;
      if (mdlbit==2)
         mdlbit=0;}
    else if (FrameMode==4)
    { ManchesterDiferencialDecoder();
      mdlbit++;
      if (mdlbit==2)
         mdlbit=0;}
    else if (FrameMode==5)
    { AmiRzpDecoder();
      mdlbit++;
      if (mdlbit==2)
         mdlbit=0;}
    else if (FrameMode==6)
    { AMINrzpDecoder();
```

```
    mdlbit=0;}
   else if (FrameMode==7)
   { DiferencialMarcaDecoder();
    mdlbit=0;}
   else if (FrameMode==8)
   { DiferencialEspacioDecoder();
    mdlbit=0;}
  if ((framecounter==0)&&(startbit==1))
  {  SetFrequency[k]=SampleTime;
    k++;
    if (k>7)
      k=0;
  }
  if (framebitcounter==6)
  {  if (framesyncr==0)
    {  framesyncr=1;
      frametotal=0;
      for (k=1;k<=5;++k)
      {  Frametotal=SetFrequency[k]+Frametotal;}
      Frametotal=Frametotal/5;
      FrameTime=FrameTotal+90;
      set_timer1(65535-frametime);
    }
  }
   if (framebitcounter>7)
   { if (mdlbit==0)
    {  framebitcounter=0;
      if (framecounter<12)
      {  Framedata[framecounter]=datarx;
        datarx=0;
        if ((framecounter>1)&&(Framecounter>=(FrameData[1]+1)))
        {  FrameCounter=0;
          FrameEnd=1;
          SETUP_TIMER_1(T1_disabled);} // timer 1 apagado}
        else
        {  Framecounter++;
        //   FrameEnd=0;
        }
      }
      else
      {  datarx=0;
        frameend=1;
        set_timer1(0);
        setup_timer_1(T1_disabled);
        framecounter=0;}
      }
   }
   finrbint:
   z=0;
}
```

```
// interrupcion de timer 0
#int_timer1
void timer1interrutpt(void)
{
  clear_interrupt(int_timer1);
  set_timer1(65535-frametime);
  if (framesyncr>0)
  {
    changes= input_b() & 0b00110000;
   if (frameBit==0)
   { FrameBit=0b00000001;}
   if (framemode==1)
   { NRZUDecoder();
     mdlbit==0;}
   else if (FrameMode==2)
   { NRZPDecoder();
     mdlbit=0;}
   else if (FrameMode==3)
   { ManchesterDecoder();
     mdlbit++;
     if (mdlbit==2)
       mdlbit=0;}
   else if (FrameMode==4)
   { ManchesterDiferencialDecoder();
     mdlbit++;
     if (mdlbit==2)
       mdlbit=0;
   }
   else if (FrameMode==5)
   { AmiRzpDecoder();
     mdlbit++;
     if (mdlbit==2)
       mdlbit=0;}
   else if (FrameMode==6)
   { AMINrzpDecoder();
     mdlbit=0;}
   else if (FrameMode==7)
   { DiferencialMarcaDecoder();
     mdlbit=0;}
   else if (FrameMode==8)
   { DiferencialEspacioDecoder();
     mdlbit=0;}
   if (framebitcounter>7)
   { if (mdlbit==0)
     {  framebitcounter=0;
       if (framecounter<12)
       {  Framedata[framecounter]=datarx;
          datarx=0;
          if ((framecounter>1)&&(Framecounter>=(FrameData[1]+1)))
          {  FrameCounter=0;
```

```
            FrameEnd=1;

            SETUP_TIMER_1(T1_disabled);} // timer 1 apagado}
          else
          {  Framecounter++;
          //   FrameEnd=0;
          }
        }
        else
        {  datarx=0;
          frameend=1;
          set_timer1(0);
          setup_timer_1(T1_disabled);
          framecounter=0;
        }
      }
    }
   }
}


//////////////////////////////////////////////////////////////////////////
//
// usb_debug_task()
//
// When called periodically, displays debugging information over serial
// to display enumeration and connection states.  Also lights LED1 based upon
// enumeration and status.
//
//////////////////////////////////////////////////////////////////////////
void usb_debug_task(void)
{
  static int8 last_connected;
  static int8 last_enumerated;
  int8 new_connected;
  int8 new_enumerated;
  static int8 last_cdc;
  int8 new_cdc;
  new_connected=usb_attached();
  new_enumerated=usb_enumerated();
  new_cdc=usb_cdc_connected();
  last_connected=new_connected;
  last_enumerated=new_enumerated;
  last_cdc=new_cdc;
}
// set timer 1
void settimer(void)
{  temptime=3000000;
  temptime=temptime/framefrequency;
  temptime=temptime/2;
```

```
  frametime=temptime;
}
// configura tiempo nrz framemode=1

// procesa códigos de línea

void ProcessFrame(void)
{

}

void MainScreen(void)
{

}
// procesa dato entrada mediante puerto usb
void ShowFrame(void)
{
  if (FrameMode==1)
  {  lcd_putc('\f');
     lcd_putc("NRZu");}
  else if (FrameMode==2)
  {  lcd_putc('\f');
     lcd_putc("NRZp");}
  else if (FrameMode==3)
  {  lcd_putc('\f');
     lcd_putc("Manchester");}
  else if (FrameMode==4)
  {  lcd_putc('\f');
     lcd_putc("Manch. Dif");}
  else if (FrameMode==5)
  {  lcd_putc('\f');
     lcd_putc("AMIRZP");}
  else if (FrameMode==6)
  {  lcd_putc('\f');
     lcd_putc("AMINRZP");}
  else if (FrameMode==7)
  {  lcd_putc('\f');
     lcd_putc("Dif. Marca");}
  else if (FrameMode==8)
  {  lcd_putc('\f');
     lcd_putc("Dif. Espacio");}
}
void Processcommand(void)
{
  if ((datainput[2]=='T')&(datainput[3]=='I')) // comando trama codigo de linea
  {  FrameMode=DataInput[4]; // tipo codigo linea a codificar 1,2,.....
     ShowFrame();
  }
}
```

```
void ReceiveData(void)
{  c=usb_cdc_getc(); //lectura datos puerto serial
//   lcd_putc(idata+0x30);
   if ((c==0xfe)&&(Ftrama==0)) // peticion nueva trama
   {  idata=0;
      length=0;
      Ftrama=1;}
   if (Ftrama==1)
   {  if (idata==1)
      {  length=c;}
        datainput[idata]=c;
      if (idata>=(length-1))
      {  Ftrama=0;
        Processcommand();}
  }
    idata++;
}
void ResetBuffers(void)
{  for (i=0;i<=16;++i)
   {  framedata[i]=0;}

}
void ResetFrame(void)
{  changesTemp= input_b() & 0b00110000;
   clear_interrupt(int_timer1);
   clear_interrupt(int_timer2);
   set_timer1(0);
   SETUP_TIMER_1(T1_disabled); // timer 1 apagado
   setup_timer_2(T2_disabled,150,1);
   Framecounter=0;
   Framebitcounter=0;
   DataRX=0;
   k=0;
   startbit=0;
   FrameEnd=0;
   Frametime=0;
   framesyncr=0;
   mdlbit=0;
   ftimer2delay=0;
   framebit=0;
}
void DecimalToHex(unsigned int8 DataInput)
{  Temp2=DataInput/0x10;
   Temp1=DataInput-Temp2*0x10;
   if (Temp1<10)
   {  Temp1=Temp1+0x30;
   }
   else
   {  Temp1=temp1+55;}
```

```
   if (Temp2<10)
   {  Temp2=Temp2+0x30;
   }
   else
   {  Temp2=temp2+55;}
}

void main(void)
{  setup_adc_ports(NO_ANALOGS);
   setup_adc(ADC_CLOCK_DIV_2);
   set_tris_b (0xff);
   ftrama=0;
   idata=0;
   FrameMode=1;
   usb_init_cs();
   lcd_init();
   delay_ms(8);
   lcd_putc('\f');
   lcd_putc("DECODIFICADOR");
   lcd_gotoxy(1,2);
   lcd_putc(FRAMEMODE+0X30);
   delay_ms(100);
   ShowFrame();
   resetbuffers();
   resetframe();
   set_timer1(0);
   SETUP_TIMER_1(T1_DIV_BY_8|T1_INTERNAL); // configura al timer 1 como
temporizador y preescaler 8
   SETUP_TIMER_1(T1_disabled); // timer 1 apagado
   setup_timer_2(T2_DIV_BY_4,150,1);
   setup_timer_2(T2_disabled,150,1);
   enable_interrupts(int_timer1); // habilita interrupcion del timer1
   enable_interrupts(int_timer2); // habilita interrupcion del timer1
   clear_interrupt(int_timer1);
   clear_interrupt(int_timer2);
   changesTemp= input_b() & 0b00110000;
   enable_interrupts(INT_RB);
   enable_interrupts(global);
   while (TRUE)
   {  usb_task();
      if (usb_cdc_kbhit()) // recepcion datos usb cdc puerto serial
      {  Receivedata();
      }
      if (Button_Frame()==1) // aumenta frecuencia
      {  delay_ms(300);
         framemode++;
         if (framemode>9)
            framemode=1;
         ShowFrame();
      }
```

```
      if (Button_Reset()==1)
      { ResetFrame();
        lcd_putc('\f');
        delay_ms(300);
        ShowFrame();
      }
      if (FrameEnd==1)
      {  lcd_putc('\f');
         ShowFrame();
         delay_ms(300);

         frameend=0;
         ResetFrame();
         lcd_gotoxy(1,2);
         if ((framedata[1]>0)&&(framedata[1]<9))
         {
           for (i=2;i<=(FrameData[1]+1);++i)
           {  DecimalToHex(FrameData[i]);
              lcd_putc(Temp2);
              lcd_putc(Temp1);
           }
           usb_task();
           if (usb_enumerated()==1) // =1 si encuentra conectado
           {  usb_cdc_putc('S');
              for (i=0;i<=8;++i)
              {  usb_cdc_putc(framedata[i+1]);}
           }
         }
      }
    }
}
```

ANEXO 6

LISTA DE MATERIALES

ANEXO # 6

Lista de Materiales Utilizados para la elaboración de los módulos

Lista de Materiales Módulo Codificador

Fuente:

- 2 diodos 1N4007.
- 2 Capacitores electrolíticos 2200 uF.
- 1 LM7805
- 1 LM7905
- 2 Capacitores cerámicos 104

Codificador

- 1 PIC 18F4550
- 2 Capacitores cerámicos 22pF.
- 2 Capacitores cerámicos de 0.1 uF.
- 1 Multiplexor CD4052BC
- 2 Resistencias de 100 KΩ
- 1 Resistencia de 10 KΩ.
- 1 Teclado
- 1 LCD 16 líneas

Lista de Materiales Módulo Decodificador.

Fuente:

- 2 diodos 1N4007.
- 2 Capacitores electrolíticos 2200 uF.
- 1 LM7805
- 1 LM7905
- 2 Capacitores cerámicos 104

Decodificador

- 1 PIC 18F4550
- 2 Capacitores cerámicos 22pF.
- 2 Capacitores cerámicos de 0.1 uF.
- 1 Multiplexor CD4052BC
- 2 Resistencias de 100 KΩ
- 1 Resistencia de 10 KΩ.
- 3 Pulsadores
- 1 LCD 16 líneas

PIC18F2455/2550/4455/4550

28/40/44-Pin High-Performance, Enhanced Flash USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant SIE
- Low-speed (1.5 Mb/s) and full-speed (12 Mb/s)
- Supports control, interrupt, isochronous and bulk transfers
- Supports up to 32 endpoints (16 bidirectional)
- 1-Kbyte dual access RAM for USB
- On-board USB transceiver with on-chip voltage regulator
- Interface for off-chip USB transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 μA typical
- Sleep current down to 0.1 μA typical
- Timer1 oscillator: 1.1 μA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 μA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Five Crystal modes, including High-Precision PLL for USB
- Two External RC modes, up to 4 MHz
- Two External Clock modes, up to 40 MHz
- Internal oscillator block:
  - 8 user selectable frequencies, from 31 kHz to 8 MHz
  - User tunable to compensate for frequency drift
- Secondary oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High current sink/source: 25 mA/25 mA
- Three external interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 6.25 ns (TCY/16)
  - Compare is 16-bit, max. resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead-time
  - Auto-Shutdown and Auto-Restart
- Addressable USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI™ (all 4 modes) and I^2C™ Master and Slave modes
- 10-bit, up to 13-channels Analog-to-Digital Converter module (A/D) with programmable acquisition time
- Dual analog comparators with input multiplexing

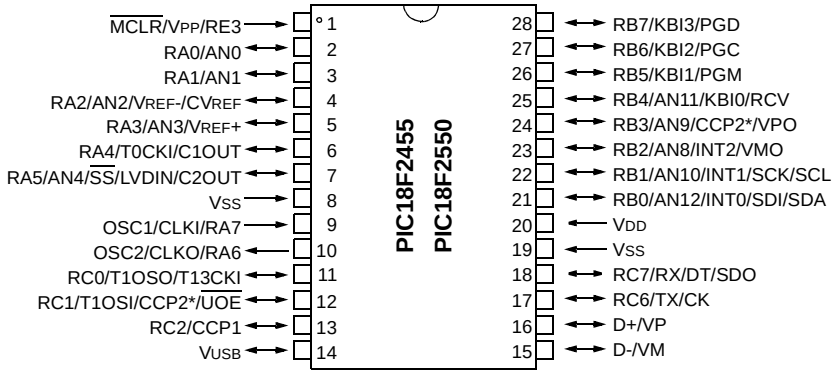
Special Microcontroller Features:

- C compiler optimized architecture with optional extended instruction set
- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle data EEPROM memory typical
- Flash/data EEPROM retention: > 40 years
- Self-programmable under software control
- Priority levels for interrupts
- 8 x 8 Single Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Wide operating voltage range (2.0V to 5.5V)

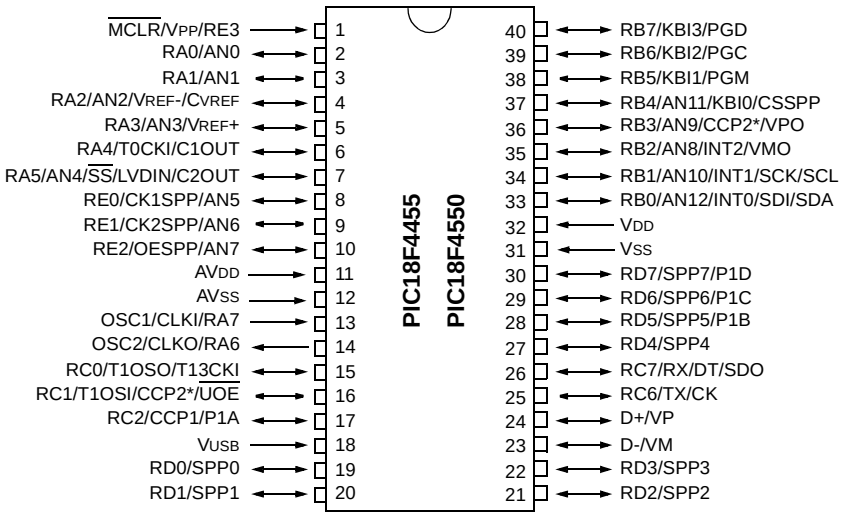
Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP/ ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-bit
	FLASH (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI	Master I^2C			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

Pin Diagrams

28-Pin SDIP, SOIC



40-Pin PDIP

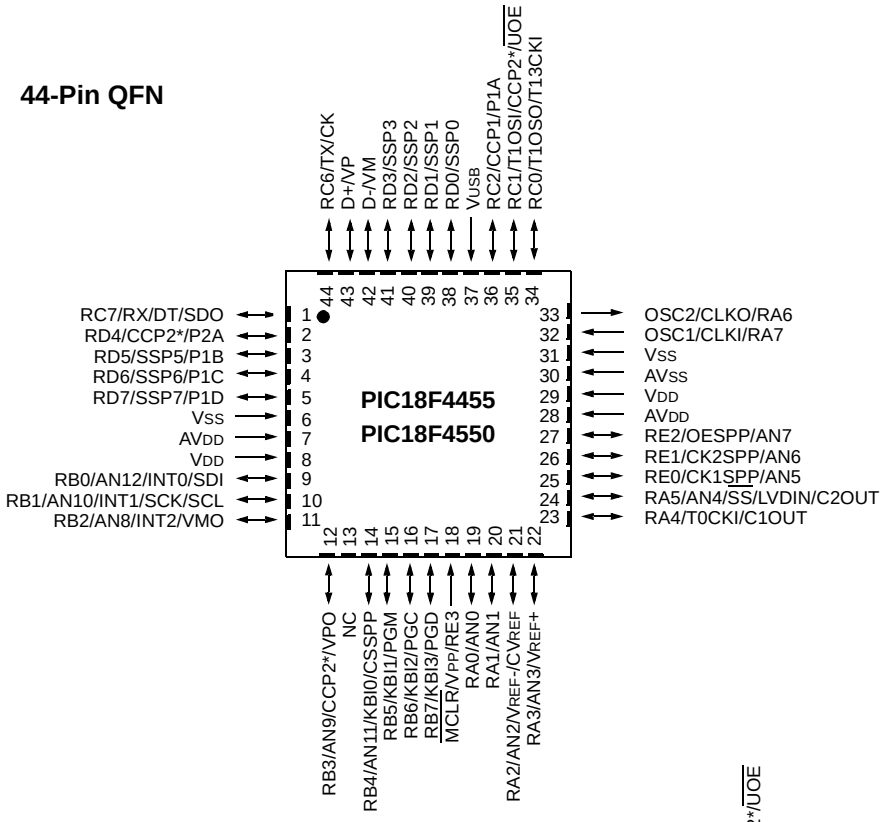


Note: Pinouts are subject to change.

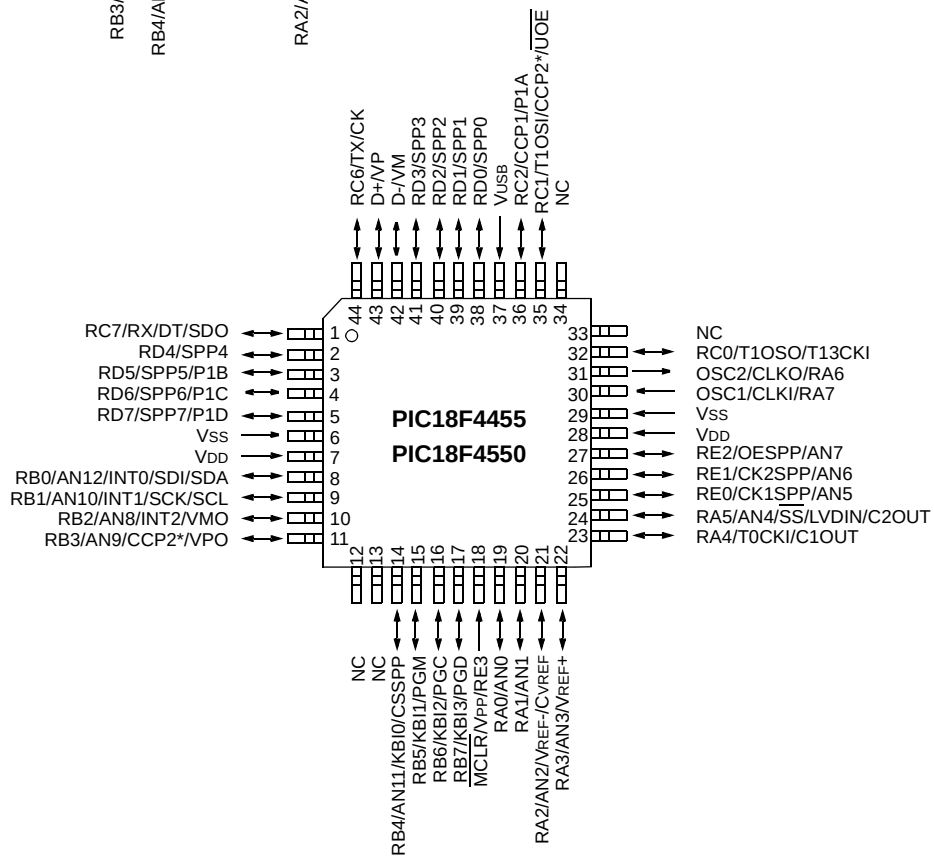
* Assignment of this feature is dependent on device configuration.

Pin Diagrams (Continued)

44-Pin QFN



44-Pin TQFP



Note: Pinouts are subject to change.

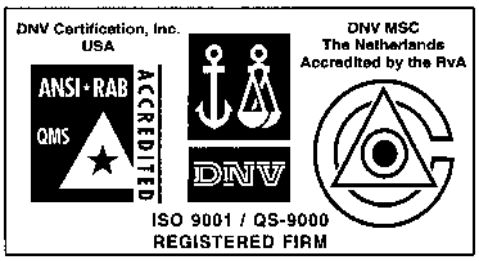
* Assignment of this feature is dependent on device configuration.

NOTES:

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Atlanta
3780 Mansell Road, Suite 130
Alpharetta, GA 30022
Tel: 770-640-0034
Fax: 770-640-0307

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848
Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
2767 S. Albright Road
Kokomo, IN 46902
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888
Fax: 949-263-1338

Phoenix
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966
Fax: 480-792-4338

San Jose
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950
Fax: 408-436-7955

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Australia
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100
Fax: 86-10-85282104

China - Chengdu
Rm. 2401-2402, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-86766200
Fax: 86-28-86766599

China - Fuzhou
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506
Fax: 86-591-7503521

China - Hong Kong SAR
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Shanghai
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700
Fax: 86-21-6275-5060

China - Shenzhen
Rm. 1812, 18/F, Building A, United Plaza
No. 5022 Binhe Road, Futian District
Shenzhen 518033, China
Tel: 86-755-82901380
Fax: 86-755-8295-1393

China - Shunde
Room 401, Hongjian Building
No. 2 Fengxiangnan Road, Ronggui Town
Shunde City, Guangdong 528303, China
Tel: 86-765-8395507 Fax: 86-765-8395571

China - Qingdao
Rm. B505A, Fullhope Plaza,
No. 12 Hong Kong Central Rd.
Qingdao 266071, China
Tel: 86-532-5027355 Fax: 86-532-5027205

India
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5932 or
82-2-558-5934

Singapore
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-6334-8870 Fax: 65-6334-8850

Taiwan
Kaohsiung Branch
30F - 1 No. 8
Min Chuan 2nd Road
Kaohsiung 806, Taiwan
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan
Taiwan Branch
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Austria
Durisolstrasse 2
A-4600 Wels
Austria
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

Denmark
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45-4420-9895 Fax: 45-4420-9910

France
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany
Steinheilstrasse 10
D-85737 Ismaning, Germany
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy
Via Quasimodo, 12
20025 Legnano (MI)
Milan, Italy
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands
P. A. De Biesbosch 14
NL-5152 SC Drunen, Netherlands
Tel: 31-416-690399
Fax: 31-416-690340

United Kingdom
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44-118-921-5869
Fax: 44-118-921-5820

07/28/03

National Semiconductor

October 1989

CD4051BM/CD4051BC Single 8-Channel Analog Multiplexer/Demultiplexer
CD4052BM/CD4052BC Dual 4-Channel Analog Multiplexer/Demultiplexer
CD4053BM/CD4053BC Triple 2-Channel Analog Multiplexer/Demultiplexer

General Description

These analog multiplexers/demultiplexers are digitally controlled analog switches having low ''ON'' impedance and very low ''OFF'' leakage currents. Control of analog signals up to $15V_{p-p}$ can be achieved by digital signal amplitudes of 3–15V. For example, if $V_{DD}=5V$, $V_{SS}=0V$ and $V_{EE}=-5V$, analog signals from $-5V$ to $+5V$ can be controlled by digital inputs of 0–5V. The multiplexer circuits dissipate extremely low quiescent power over the full $V_{DD}-V_{SS}$ and $V_{DD}-V_{EE}$ supply voltage ranges, independent of the logic state of the control signals. When a logical ''1'' is present at the inhibit input terminal all channels are ''OFF''.

CD4051BM/CD4051BC is a single 8-channel multiplexer having three binary control inputs. A, B, and C, and an inhibit input. The three binary signals select 1 of 8 channels to be turned ''ON'' and connect the input to the output.

CD4052BM/CD4052BC is a differential 4-channel multiplexer having two binary control inputs, A and B, and an inhibit input. The two binary input signals select 1 or 4 pairs of channels to be turned on and connect the differential analog inputs to the differential outputs.

CD4053BM/CD4053BC is a triple 2-channel multiplexer having three separate digital control inputs, A, B, and C, and an inhibit input. Each control input selects one of a pair of channels which are connected in a single-pole double-throw configuration.

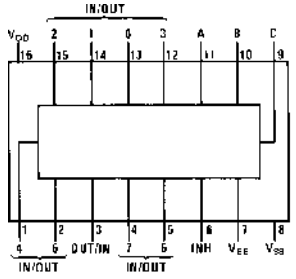
Features

- Wide range of digital and analog signal levels: digital 3–15V, analog to $15V_{p-p}$
- Low ''ON'' resistance: 80Ω (typ.) over entire $15V_{p-p}$ signal-input range for $V_{DD}-V_{EE}=15V$
- High ''OFF'' resistance: channel leakage of ±10 pA (typ.) at $V_{DD}-V_{EE}=10V$
- Logic level conversion for digital addressing signals of 3–15V ($V_{DD}-V_{SS}=3-15V$) to switch analog signals to 15 V_{p-p} ($V_{DD}-V_{EE}=15V$)
- Matched switch characteristics: $\Delta R_{ON}=5\Omega$ (typ.) for $V_{DD}-V_{EE}=15V$
- Very low quiescent power dissipation under all digital-control input and supply conditions: 1 μW (typ.) at $V_{DD}-V_{SS}=V_{DD}-V_{EE}=10V$
- Binary address decoding on chip

Connection Diagrams

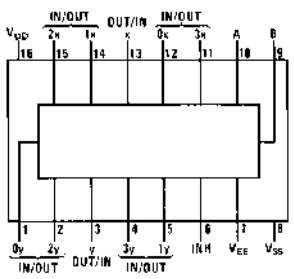
Dual-In-Line Packages

CD4051BM/CD4051BC



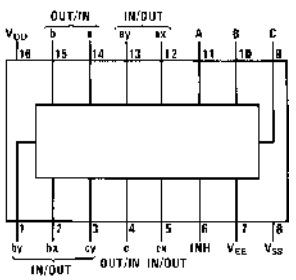
TOP VIEW

CD4052BM/CD4052BC



TOP VIEW

CD4053BM/CD4053BC



TOP VIEW

TL/F/5662–1

Order Number CD4051B, CD4052B, or CD4053B

RRD-B30M105/Printed in U. S. A.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

DC Supply Voltage (V_{DD})	$-0.5\ V_{DC}$ to $+18\ V_{DC}$
Input Voltage (V_{IN})	$-0.5\ V_{DC}$ to $V_{DD}+0.5\ V_{DC}$
Storage Temperature Range (T_S)	$-65°C$ to $+150°C$
Power Dissipation (P_D)	
Dual-In-Line	700 mW
Small Outline	500 mW
Lead Temp. (T_L) (soldering, 10 sec.)	260°C

Recommended Operating Conditions

DC Supply Voltage (V_{DD})	$+5\ V_{DC}$ to $+15\ V_{DC}$
Input Voltage (V_{IN})	0V to $V_{DD}\ V_{DC}$
Operating Temperature Range (T_A)	
4051BM/4052BM/4053BM	$-55°C$ to $+125°C$
4051BC/4052BC/4053BC	$-40°C$ to $+85°C$

DC Electrical Characteristics (Note 2)

Symbol	Parameter	Conditions		−55°C		+25°			+125°C		Units
				Min	Max	Min	Typ	Max	Min	Max	
I_{DD}	Quiescent Device Current	$V_{DD}=5V$ $V_{DD}=10V$ $V_{DD}=15V$			5 10 20			5 10 20		150 300 600	μA μA μA
Signal Inputs (V_{IS}) and Outputs (V_{OS})											
R_{ON}	"ON" Resistance (Peak for $V_{EE} \le V_{IS} \le V_{DD}$)	$R_L=10\ k\Omega$ (any channel selected)	$V_{DD}=2.5V$, $V_{EE}=-2.5V$ or $V_{DD}=5V$, $V_{EE}=0V$		800		270	1050		1300	Ω
			$V_{DD}=5V$ $V_{EE}=-5V$ or $V_{DD}=10V$, $V_{EE}=0V$		310		120	400		550	Ω
			$V_{DD}=7.5V$, $V_{EE}=-7.5V$ or $V_{DD}=15V$, $V_{EE}=0V$		200		80	240		320	Ω
ΔR_{ON}	Δ"ON" Resistance Between Any Two Channels	$R_L=10\ k\Omega$ (any channel selected)	$V_{DD}=2.5V$, $V_{EE}=-2.5V$ or $V_{DD}=5V$, $V_{EE}=0V$				10				Ω
			$V_{DD}=5V$, $V_{EE}=-5V$ or $V_{DD}=10V$, $V_{EE}=0V$				10				Ω
			$V_{DD}=7.5V$, $V_{EE}=-7.5V$ or $V_{DD}=15V$, $V_{EE}=0V$				5				Ω
	"OFF" Channel Leakage Current, any channel "OFF"	$V_{DD}=7.5V$, $V_{EE}=-7.5V$ $O/I=\pm 7.5V$, $I/O=0V$			± 50		± 0.01	± 50		± 500	nA
	"OFF" Channel Leakage Current, all channels "OFF" (Common OUT/IN)	Inhibit = 7.5V $V_{DD}=7.5V$, $V_{EE}=-7.5V$, $O/I=0V$, $I/O=\pm 7.5V$	CD4051 CD4052 CD4053		± 200 ± 200 ± 200		± 0.08 ± 0.04 ± 0.02	± 200 ± 200 ± 200		± 2000 ± 2000 ± 2000	nA nA nA
Control Inputs A, B, C and Inhibit											
V_{IL}	Low Level Input Voltage	$V_{EE}=V_{SS}$ $R_L=1\ k\Omega$ to V_{SS} $I_{IS}<2\ \mu A$ on all OFF channels $V_{IS}=V_{DD}$ thru $1\ k\Omega$ $V_{DD}=5V$ $V_{DD}=10V$ $V_{DD}=15V$			1.5 3.0 4.0			1.5 3.0 4.0		1.5 3.0 4.0	V V V
V_{IH}	High Level Input Voltage	$V_{DD}=5$ $V_{DD}=10$ $V_{DD}=15$		3.5 7 11		3.5 7 11			3.5 7 11		V V V

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. Except for "Operating Temperature Range" they are not meant to imply that the devices should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All voltages measured with respect to V_{SS} unless otherwise specified.

DC Electrical Characteristics (Note 2) (Continued)

Symbol	Parameter	Conditions	−40°C		+25°C			+85°C		Units
			Min	Max	Min	Typ	Max	Min	Max	
I_{IN}	Input Current	$V_{DD}=15V$, $V_{EE}=0V$ $V_{IN}=0V$		−0.1		-10^{-5}	−0.1		−1.0	μA
		$V_{DD}=15V$, $V_{EE}=0V$ $V_{IN}=15V$		0.1		10^{-5}	0.1		1.0	μA
I_{DD}	Quiescent Device Current	$V_{DD}=5V$		20			20		150	μA
		$V_{DD}=10V$		40			40		300	μA
		$V_{DD}=15V$		80			80		600	μA
Signal Inputs (V_{IS}) and Outputs (V_{OS})										
R_{ON}	"ON" Resistance (Peak for $V_{EE} \le V_{IS} \le V_{DD}$)	$R_L=10\ k\Omega$ (any channel selected); $V_{DD}=2.5V$, $V_{EE}=-2.5V$ or $V_{DD}=5V$, $V_{EE}=0V$		850		270	1050		1200	Ω
		$V_{DD}=5V$, $V_{EE}=-5V$ or $V_{DD}=10V$, $V_{EE}=0V$		330		120	400		520	Ω
		$V_{DD}=7.5V$, $V_{EE}=-7.5V$ or $V_{DD}=15V$, $V_{EE}=0V$		210		80	240		300	Ω
ΔR_{ON}	Δ"ON" Resistance Between Any Two Channels	$R_L=10\ k\Omega$ (any channel selected); $V_{DD}=2.5V$, $V_{EE}=-2.5V$ or $V_{DD}=5V$, $V_{EE}=0V$				10				Ω
		$V_{DD}=5V$ $V_{EE}=-5V$ or $V_{DD}=10V$, $V_{EE}=0V$				10				Ω
		$V_{DD}=7.5V$, $V_{EE}=-7.5V$ or $V_{DD}=15V$, $V_{EE}=0V$				5				Ω
	"OFF" Channel Leakage Current, any channel "OFF"	$V_{DD}=7.5V$, $V_{EE}=-7.5V$ $O/I=\pm7.5V$, $I/O=0V$		±50		±0.01	±50		±500	nA
	"OFF" Channel Leakage Current, all channels "OFF" (Common OUT/IN)	Inhibit = 7.5V CD4051 $V_{DD}=7.5V$, $V_{EE}=-7.5V$, CD4052 $O/I=0V$ $I/O=\pm7.5V$ CD4053		±200 ±200 ±200		±0.08 ±0.04 ±0.02	±200 ±200 ±200		±2000 ±2000 ±2000	nA nA nA
Control Inputs A, B, C and Inhibit										
V_{IL}	Low Level Input Voltage	$V_{EE}=V_{SS}$ $R_L=1\ k\Omega$ to V_{SS} $I_{IS}<2\ \mu A$ on all OFF Channels $V_{IS}=V_{DD}$ thru 1 kΩ								
		$V_{DD}=5V$		1.5			1.5		1.5	V
		$V_{DD}=10V$		3.0			3.0		3.0	V
		$V_{DD}=15V$		4.0			4.0		4.0	V
V_{IH}	High Level Input Voltage	$V_{DD}=5$	3.5		3.5			3.5		V
		$V_{DD}=10$	7		7			7		V
		$V_{DD}=15$	11		11			11		V
I_{IN}	Input Current	$V_{DD}=15V$, $V_{EE}=0V$ $V_{IN}=0V$		−0.1		-10^{-5}	−0.1		−1.0	μA
		$V_{DD}=15V$, $V_{EE}=0V$ $V_{IN}=15V$		0.1		10^{-5}	0.1		1.0	μA

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. Except for "Operating Temperature Range" they are not meant to imply that the devices should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All voltages measured with respect to V_{SS} unless otherwise specified.

AC Electrical Characteristics* $T_A = 25°C$, $t_r = t_f = 20$ ns, unless otherwise specified.

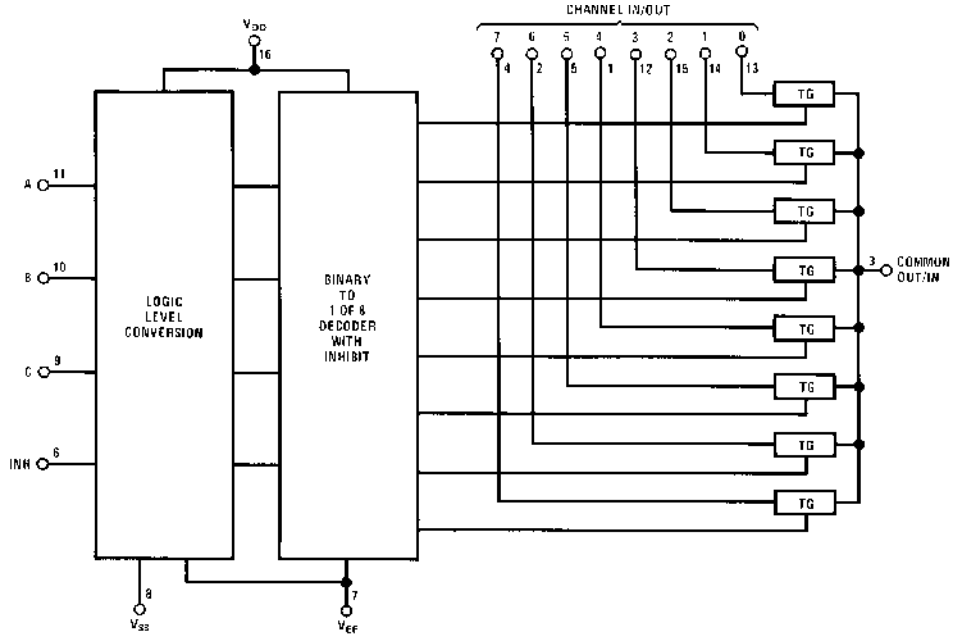
Symbol	Parameter	Conditions	V_{DD}	Min	Typ	Max	Units
t_{PZH}, t_{PZL}	Propagation Delay Time from Inhibit to Signal Output (channel turning on)	$V_{EE} = V_{SS} = 0V$ $R_L = 1\ k\Omega$ $C_L = 50$ pF	5V 10V 15V		600 225 160	1200 450 320	ns ns ns
t_{PHZ}, t_{PLZ}	Propagation Delay Time from Inhibit to Signal Output (channel turning off)	$V_{EE} = V_{SS} = 0V$ $R_L = 1\ k\Omega$ $C_L = 50$ pF	5V 10V 15V		210 100 75	420 200 150	ns ns ns
C_{IN}	Input Capacitance Control input Signal Input (IN/OUT)				 5 10	 7.5 15	 pF pF
C_{OUT}	Output Capacitance (common OUT/IN)						
	CD4051 CD4052 CD4053	$V_{EE} = V_{SS} = 0V$	10V 10V 10V		30 15 8		pF pF pF
C_{IOS}	Feedthrough Capacitance				0.2		pF
C_{PD}	Power Dissipation Capacitance						
	CD4051 CD4052 CD4053				110 140 70		pF pF pF
Signal Inputs (V_{IS}) and Outputs (V_{OS})							
	Sine Wave Response (Distortion)	$R_L = 10\ k\Omega$ $f_{IS} = 1$ kHz $V_{IS} = 5\ V_{p-p}$ $V_{EE} = V_{SI} = 0V$	10V		0.04		%
	Frequency Response, Channel "ON" (Sine Wave Input)	$R_L = 1\ k\Omega$, $V_{EE} = 0V$, $V_{IS} = 5V_{p-p}$, $20 \log_{10} V_{OS}/V_{IS} = -3$ dB	10V		40		MHz
	Feedthrough, Channel "OFF"	$R_L = 1\ k\Omega$, $V_{EE} = V_{SS} = 0V$, $V_{IS} = 5V_{p-p}$, $20 \log_{10} V_{OS}/V_{IS} = -40$ dB	10V		10		MHz
	Crosstalk Between Any Two Channels (frequency at 40 dB)	$R_L = 1\ k\Omega$, $V_{EE} = V_{SS} = 0V$, $V_{IS}(A) = 5V_{p-p}$ $20 \log_{10} V_{OS}(B)/V_{IS}(A) = -40$ dB (Note 3)	10V		3		MHz
t_{PHL} t_{PLH}	Propagation Delay Signal Input to Signal Output	$V_{EE} = V_{SS} = 0V$ $C_L = 50$ pF	5V 10V 15V		25 15 10	55 35 25	ns ns ns
Control Inputs, A, B, C and Inhibit							
	Control Input to Signal Crosstalk	$V_{EE} = V_{SS} = 0V$, $R_L = 10\ k\Omega$ at both ends of channel. Input Square Wave Amplitude = 10V	10V		65		mV (peak)
t_{PHL}, t_{PLH}	Propagation Delay Time from Address to Signal Output (channels "ON" or "OFF")	$V_{EE} = V_{SS} = 0V$ $C_L = 50$ pF	5V 10V 15V		500 180 120	1000 360 240	ns ns ns

*AC Parameters are guaranteed by DC correlated testing.

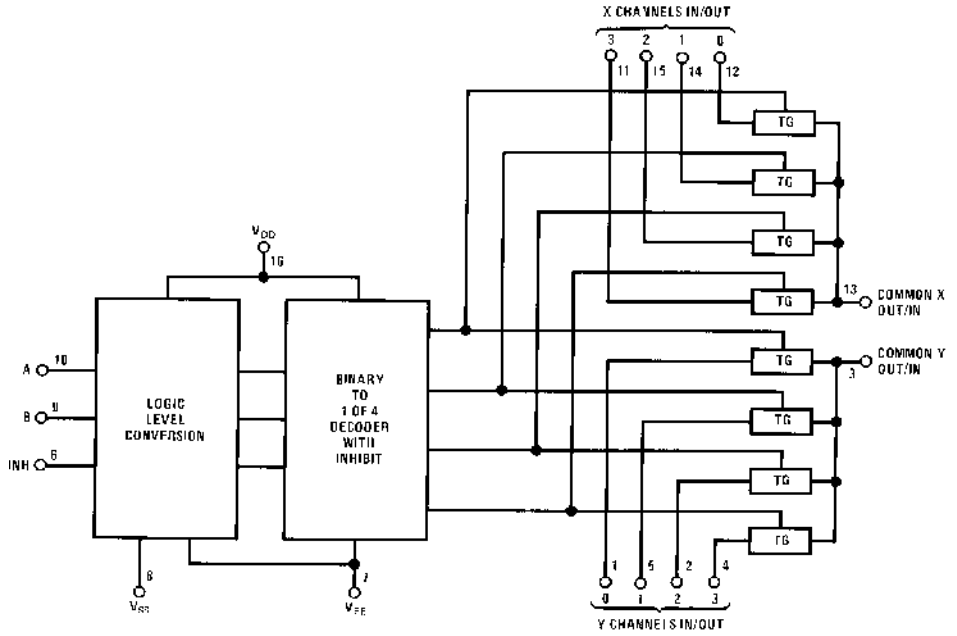
Note 3: A, B are two arbitrary channels with A turned "ON" and B "OFF".

Block Diagrams

CD4051BM/CD4051BC



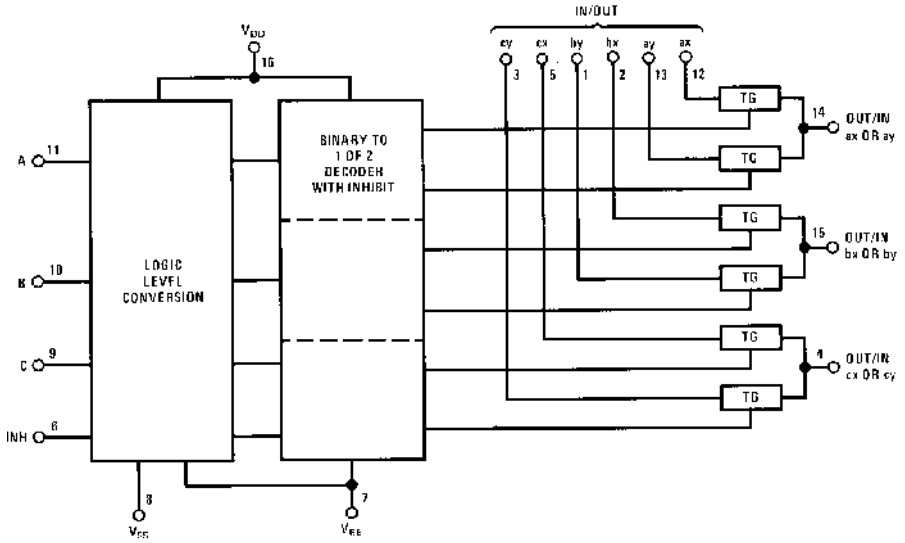
CD4052BM/CD4052BC



TL/F/5662–2

Block Diagrams (Continued)

CD4053BM/CD4053BC



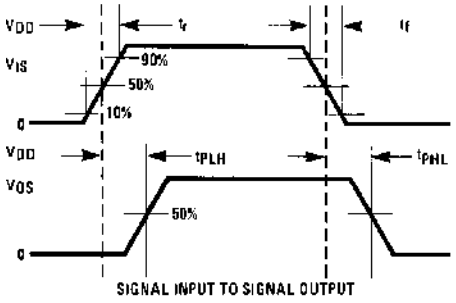
TL/F/5662–3

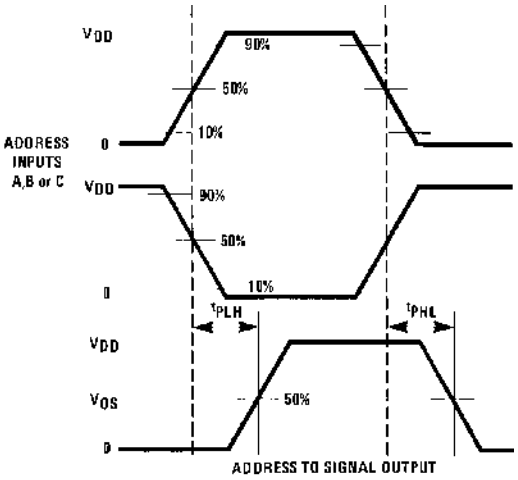
Truth Table

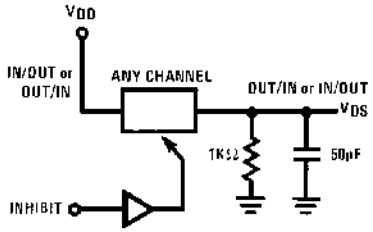
INPUT STATES				"ON" CHANNELS		
INHIBIT	**C**	**B**	**A**	**CD4051B**	**CD4052B**	**CD4053B**
0	0	0	0	0	0X, 0Y	cx, bx, ax
0	0	0	1	1	1X, 1Y	cx, bx, ay
0	0	1	0	2	2X, 2Y	cx, by, ax
0	0	1	1	3	3X, 3Y	cx, by, ay
0	1	0	0	4		cy, bx, ax
0	1	0	1	5		cy, bx, ay
0	1	1	0	6		cy, by, ax
0	1	1	1	7		cy, by, ay
1	*	*	*	NONE	NONE	NONE

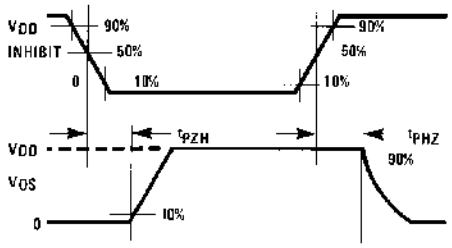
*Don't Care condition.

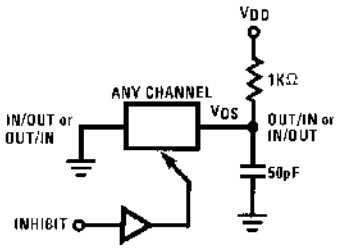
Switching Time Waveforms

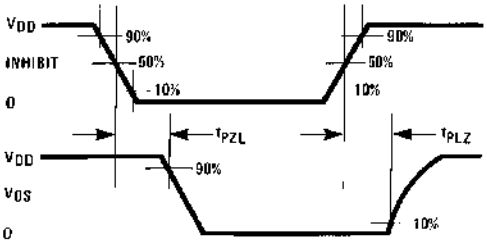












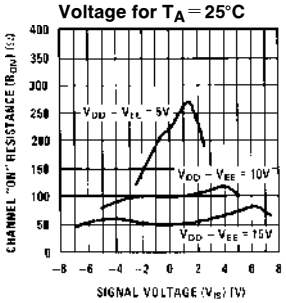
TL/F/5662–4

Special Considerations

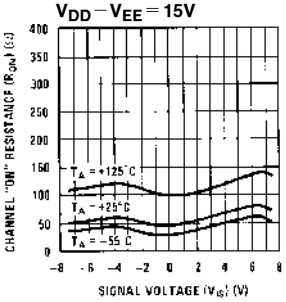
In certain applications the external load-resistor current may include both V_{DD} and signal-line components. To avoid drawing V_{DD} current when switch current flows into IN/OUT pin, the voltage drop across the bidirectional switch must not exceed 0.6V at $T_A \leq 25°C$, or 0.4V at $T_A > 25°C$ (calculated from R_{ON} values shown). No V_{DD} current will flow through R_L if the switch current flows into OUT/IN pin.

Typical Performance Characteristics

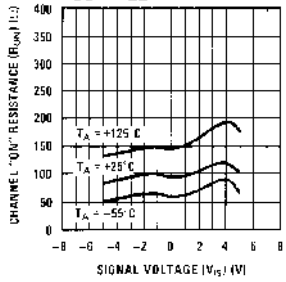
"ON" Resistance vs Signal Voltage for $T_A = 25°C$



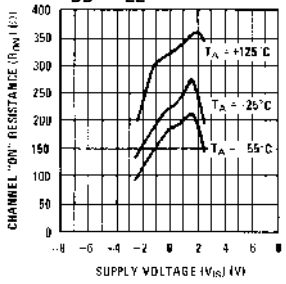
"ON" Resistance as a Function of Temperature for $V_{DD} - V_{EE} = 15V$



"ON" Resistance as a Function of Temperature for $V_{DD} - V_{EE} = 10V$

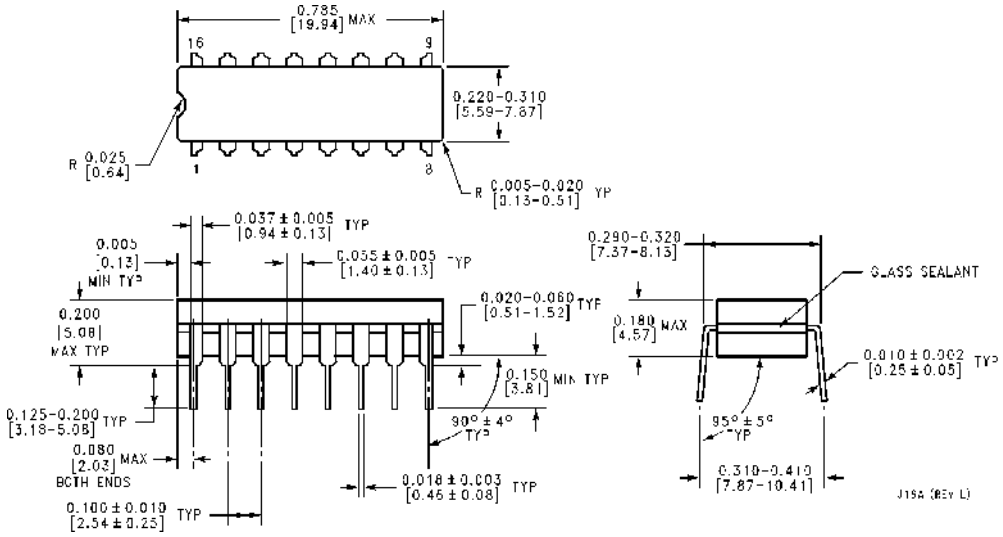


"ON" Resistance as a Function of Temperature for $V_{DD} - V_{EE} = 5V$

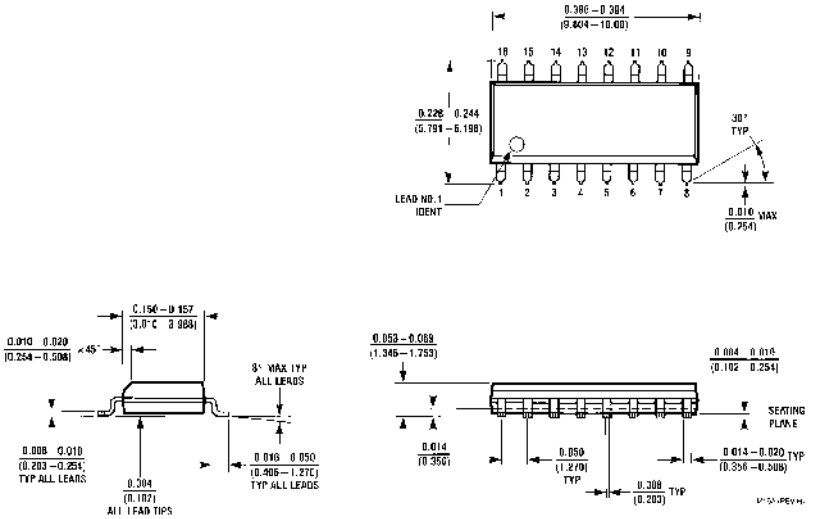


TL/F/5662–5

Physical Dimensions inches (millimeters)

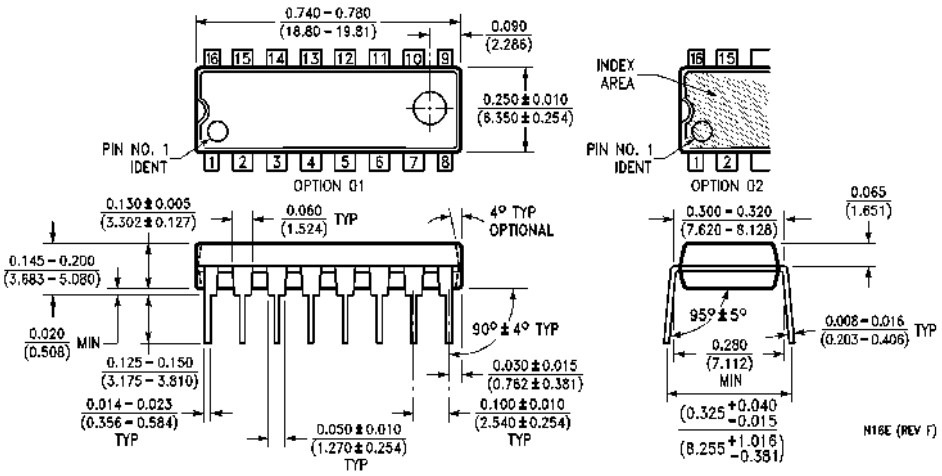


Cavity Dual-In-Line Package (J)
Order Number CD4051BMJ, CD4051BCJ, CD4052BMJ,
CD4052BCJ, CD4053BMJ or CD4053BCJ
NS Package Number J16A



Small Outline Package (M)
Order Number CD4051BCM,
CD4052BCM or CD4053BCM
NS Package Number M16A

Physical Dimensions inches (millimeters) (Continued)



Molded Dual-In-Line Package (N)
Order CD4051BM, CD4051BC,
CD4052BM, CD4052BC, CD4053BM, CD4053BC
NS Package Number N16E

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408