investigación en algún formato existente y validado por alguna revista.

Cristian David Patiño 1088334538, Jairo Alejandro Castrillón 1112782874

Módulo 7: Paper. Generación de un documento de publicación de resultados científicos y de

*Abstract* — The present project aims to develop in the student skills relevant to design, research, compliance and determination of requirements and implementation of systems embedded, as a useful part of digital systems. There is another series of non-technical skills and competences that are intended to be developed in the student as teamwork, work under pressure and in tight times, as well as the creativity and problem solving

*Index Terms* — develop, requirements, embedded, teamwork

# INTRODUCION

El presente proyecto pretende desarrollar en el estudiante las habilidades pertinentes al diseño, investigación, cumplimiento y determinación de requerimientos e implementación de sistemas embebidos, como una parte útil de los sistemas digitales.

Existen otra serie de habilidades y competencias no técnicas que se pretenden desarrollar en el estudiante como el trabajo en equipo, trabajo bajo presión y en tiempos ajustados, así como la creatividad y resolución de problemas.

# METODOLOGIA DEL CURSO

Con el fin de llevar a cabo el proyecto especificado se ha diseñado un plan de trabajo que consiste en la distribución por módulos del proyecto, cada módulo será un entregable parcial del proyecto en sí.

Cada módulo será sustentado en presencia de la totalidad del miembro del grupo de trabajo y los soportes serán entregados mediante correo electrónico al profesor (filosofía cero papel). Los archivos que se entregarán serán los archivos de código fuente (software y descripción de hardware, cual fuera el caso) y un archivo de descripción de desarrollo del módulo (Bitácora), donde se indique: ¿qué se hizo?, ¿cómo se hizo?, por qué se hizo de esa manera?, cuáles fueron las fuentes consultadas, dificultades y/o aciertos, etc.

Cada módulo tiene definido un cronograma de tiempo en semanas y un porcentaje de nota, relativo al 100% de la nota del laboratorio. Este porcentaje está definido de acuerdo al tiempo estimado de desarrollo del módulo y la dificultad del mismo.

La copia de código no está permitida como plagio. Es válido tener referentes de fuentes que tengan soluciones útiles, con la condición de que las fuentes sean relacionadas en el documento de la bitácora y que el grupo en su totalidad comprenda la solución allí implementada.

# DESCRIPCION GENERAL DEL PROYECTO

El aumento anual en el parque automotor en las ciudades del país, genera traumatismos en las arterias viales, debido a la ineficiente y poco planeada infraestructura vial. De igual forma se presenta traumatismo a la hora del parqueo, tanto para los conductores (pérdida de tiempo) como para los encargados del parqueadero (reducción de movilidad al interior del parqueadero).

Esta problemática se evidencia en grandes parqueaderos públicos, como los de los centros comerciales y clínicas.

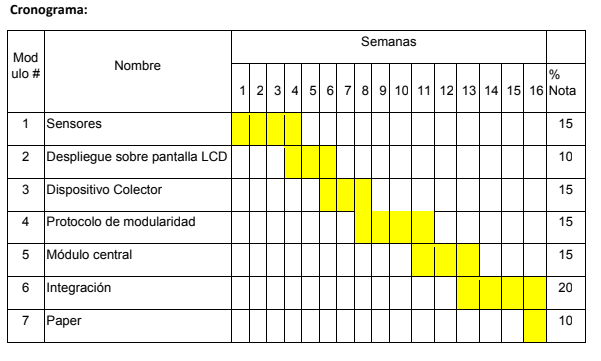
Con el fin de reducir dichos traumatismos, se pretende desarrollar un sistema de parqueo inteligente, que proporcione información en tiempo real a los conductores de la disponibilidad o no de áreas de parqueo y en qué lugares se encuentran los parqueos disponibles.

Dicho sistema estará en la capacidad de validar si hay cupos disponibles o no en el parqueadero, detectar el vehículo parqueado en un espacio determinado, que esta información (parqueadero disponible u ocupado) se envíe a un dispositivo colector para que este calcule los parqueaderos totales disponibles. el dispositivo colector está en la capacidad de desplegar la información de disponibilidad en una pantalla (Infomovilidad) y en capacidad de replicar dicha información a un dispositivo central.

El dispositivo central es utilizado por personal de seguridad del parqueadero o personal encargado de la movilidad en el parqueadero, con el fin de tomar decisiones con información precisa en todo momento.

# DESCRIPCION DETALLADA DEL PROYECTO

Fig. 2. Cronograma



Descripción detallada del proyecto:

Módulo 1: Sensores. Determinar el tipo de sensor que mejor se acople a los requerimientos del sistema e implementar un módulo que capture y despliegue la información proporcionada por el sensor.

Módulo 2: Despliegue sobre pantalla LCD. Implementar un módulo que despliegue de información de cualquier tipo sobre pantalla LCD.

Módulo 3: Dispositivo Colector. Implementación del HW necesario para la construcción de un sistema de sensado, colección de información de varios sensores y despliegue de información sobre pantalla LCD.

Módulo 4: Protocolo de modularidad. Diseñar un protocolo que permita la interconexión entre módulos colectores para administrar la información de movilidad global del parqueadero y la comunicación de la información al dispositivo central.

Módulo 5: Módulo central. Diseñar e implementar el software que permita que un computador pueda recibir la información de los dispositivos colectores.

Módulo 6: Integración de la solución. Diseños adicionales y depuración de los desarrollos previos con el fin de que se integren a la funcionalidad final del producto.

Módulo 7: Paper. Generación de un documento de publicación de resultados científicos y de investigación en algún formato existente y validado por alguna revista.

Fig. 1. Imagen que muestra el desorden para poder parquear un carro sin ayuda tecnológica e innovadora



# RESTRICCIONES

El proyecto debe ser desarrollado utilizando las tarjetas de desarrollo Basys 2 o Nexys 2, ya que serán los dispositivos disponibles en laboratorio. El profesor indicará en su debido momento cuáles serán las demás herramientas o elemento disponibles en laboratorio, sin embargo, si los estudiantes deciden utilizar otros dispositivos o herramientas, estará bajo responsabilidad del grupo la consecución de materiales adicionales para la implementación del proyecto aquí propuesto.

# MODULO 1: SENSORES

La primera parte consistió en buscar un sensor que se adaptara mejor a las necesidades que buscaba el grupo de trabajo para esto se consultaron diferentes sensores de diferentes tipos el (sensor PIR) es un sensor electrónico que mide la luz infrarroja (IR) Los sensores de distancia y transductores de distancia, están pensados para realizar la medida de distancia lineal o desplazamiento lineal de una forma automatizada, se consulto acerca de su funcionamiento y se tomó la decisión de elegir Dynadetect Detección 5-10 cm este sensor fue la elección más óptima ya que cuenta con las siguientes especificaciones:

* Voltaje Alimentación: 3-5 V.
* Compatible: Arduino, raspberry, otras.
* Tamaño: 1,2cm x 3,1cm.
* Tipo de Salida: Digital.
* Puertos: Vin, Gnd y Vout.
* Función: Detección de objetos entre 0 a 5cm.
* Además, tiene: Led de indicación frontal de detección y orificio de fijación.

Para utilizar este sensor solo tienes que conectar alimentación, tierra y la salida a tu dispositivo de control, además este sensor tiene un led en la parte frontal que indica cuando ha detectado algo.

El sensor tiene tres pines etiquetados respectivamente con Vin Gnd Vout, Vin es el voltaje de alimentación que va de 3V a 5 V, Gnd corresponde a la tierra del sistema, y Vout es la señal de salida, cuando el sensor está en color blanco el voltaje de salida es aproximadamente al voltaje de alimentación Vin y cuando está en color negro se aproxima a 0V o Gnd.

Teniendo esta información se procedió a buscar el diseño que nos permitiera simular cómo funcionaba el sensor haciendo uso de la herramienta ISE Design Suite 14.7 y de HDL- vhdl se realizó un diseño de nivel estructurado.

Vout <= not Vin;

Esta función Booleana describe muy bien el comportamiento del sensor.

La siguiente parte del diseño fue asignar los puertos de entradas digitalmente teniendo en cuenta que para este primer módulo utilizaremos la FPGA Nexys 2.

Siguiendo la guía las referencias de esta FPGA son las siguientes:

Family: Spartan 3E

Device: XC3S500C

Package: FG320

Fig. 3. Sensor a distancia 10cm en Board



Con la herramienta Xilinx PlanAhead 14.7 se le asigno los pines de entrada digitalmente a las entradas y salidas de nuestro diseño de nivel estructurado.

Vin: L14 pin de entrada

Vout: J14 pin de salida del led de FPGA

Después de asignar los pines exitosamente se procedió a generar un archivo programable y con la FPGA física a realizar las pruebas correspondientes las cuales consistían en pasar un objeto enfrente del sensor y este a su vez prender el led de la FPGA lo cual garantizaba que el diseño se había realizado con éxito.

Para este módulo se encontraron las siguientes dificultades.

* No tener conocimiento previo sobre sensores y su funcionamiento.
* La asignación de pines con el software Xilinx PlanAhead 14.7.
* A la hora de hacer la prueba con la FPGA física no se disponía si no de una con lo que tocaba seguir un orden para trabajar con esta.

# MODULO 2: DESPLIEGUE SOBRE PANTALLA LCD

Se inició una investigación por parte del equipo de trabajo del laboratorio de electrónica digital para comprender el funcionamiento de un LCD16x2 y utilizarlo para realizar el Módulo 2, como hacemos uso del protocolo HD44780U (LCD-II) para el despliegue de información en la LCD se usó una librería desarrollada por INTESC que tomaba como base este protocolo.

INTESC es una empresa mexicana dedicada al diseño de sistemas embebidos basados en FPGAs y SoCs, asimismo se dedica al desarrollo y producción de nueva tecnología en áreas de electrónica y sistemas embebidos.

Lo siguiente es explicar las funciones de que conforman la librería de nuestro código en VHDL para la LCD16X2.

por ultimo demostrar el funcionamiento del código en VHDL para la pantalla LCD16X2.

La LCD 16x2 es un componente que se utiliza para la representación de caracteres en muchos dispositivos electrónicos. El principal componente de este diseño será formado por un procesador, un paquete de comandos donde tenemos nuestras funciones de la librería y caracteres especiales, los cuales harán el trabajo de facilitarnos el uso rápido de la LCD16X2.

* Procesador: Este convierte el vector que recibe del paquete comandos en lenguaje que la LCD pueda entender
* Caracteres especiales: Es donde podremos dibujar caracteres personalizados

Las funciones comandos que se utilizan son los siguientes:

* LCD\_INI (): Inicializa la LCD.

NOTA: Dentro de los paréntesis poner un vector de 2 bits para encender o apagar el cursor y activar o desactivar el parpadeo.

"1x" -- Cursor ON

"0x" -- Cursor OFF

"x1" -- Parpadeo ON

"x0" -- Parpadeo OFF

Por ejemplo: LCD\_INI ("10") -- Inicializar LCD con cursor encendido y sin parpadeo

* CHAR (): Manda una letra mayúscula o minúscula

IMPORTANTE:

1) Debido a que VHDL no es sensible a mayúsculas y minúsculas, si se quiere escribir una letra mayúscula se debe escribir una "M" antes de la letra.

2) Si se quiere escribir la letra "S" mayúscula, se declara "MS"

Por ejemplo:

CHAR(A) -- Escribe en la LCD la letra "a"

CHAR (MA) -- Escribe en la LCD la letra "A"

CHAR(S) -- Escribe en la LCD la letra "s"

CHAR (MS) -- Escribe en la LCD la letra "S"

* POS (): Escribir en la posición que se indique.

NOTA: Dentro de los paréntesis se debe poner la posición de la LCD a la que se quiere ir, empezando por el renglón seguido de la posición vertical, ambos números separados por una coma.

Por ejemplo:

POS (1,2) -- Manda cursor al renglón 1, posición 2

POS (2,4) -- Manda cursor al renglón 2, posición 4

* CHAR\_ASCII (): Escribe un carácter a partir de su código en ASCII

NOTA: Dentro de los paréntesis escribir x"(número hex.)"

Por ejemplo: CHAR\_ASCII(x"40") -- Escribe en la LCD el carácter "@"

* CODIGO\_FIN (): Finaliza el código.

NOTA: Dentro de los paréntesis poner cualquier número: 1, 2, 3,4..., 8,9.

* BUCLE\_INI (): Indica el inicio de un bucle.

NOTA: Dentro de los paréntesis poner cualquier número: 1, 2, 3,4..., 8,9.

* BUCLE\_FIN (): Indica el final del bucle.

NOTA: Dentro de los paréntesis poner cualquier número: 1, 2, 3,4..., 8,9.

* INT\_NUM (): Escribe en la LCD un número entero.

NOTA: Dentro de los paréntesis poner sólo un número que vaya del 0 al 9,

si se quiere escribir otro número entero se tiene que volver a llamar la función

* CREAR\_CHAR (): Función que crea el carácter

"CARACTERES\_ESPECIALES.vhd"

NOTA: Dentro de los paréntesis poner el nombre del carácter dibujado (CHAR1, CHAR2, CHAR3, …, CHAR8)

* CHAR\_CREADO (): Escribe en la LCD el carácter creado por medio de la función "CREAR\_CHAR ()"

NOTA: Dentro de los paréntesis poner el nombre del carácter creado.

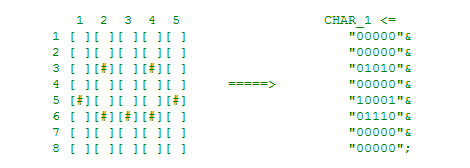
Por ejemplo:

Dentro de "CARACTERES\_ESPECIALES.vhd" se dibujan los caracteres personalizados utilizando los vectores.

"CHAR\_1", "CHAR\_2","CHAR\_3”, …,"CHAR\_7","CHAR\_8".

1 = [#] - Se activa el pixel de la matriz.

0 = [ ] - Se desactiva el pixel de la matriz.



CREAR\_CHAR(CHAR1) -- Crea el carácter personalizado (CHAR1).

CHAR\_CREADO(CHAR1) -- Muestra en la LCD el carácter creado (CHAR1).

* LIMPIAR\_PANTALLA (): Manda a limpiar la LCD.

NOTA: Ésta función se activa poniendo dentro de los paréntesis, un '1' y se desactiva con un '0'.

Ejemplo:

LIMPIAR\_PANTALLA ('1') -- Limpiar pantalla está activado

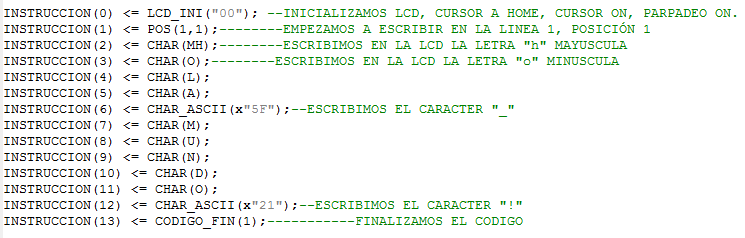
LIMPIAR\_PANTALLA ('0') -- Limpiar pantalla está desactivado.

Las entradas "CORD" y "CORI" se hacen corrimientos a la derecha y a la Izquierda respectivamente.

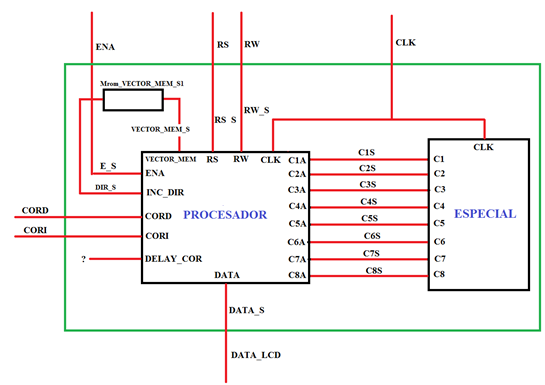
NOTA: La velocidad del corrimiento se puede cambiar modificando la variable "DELAY\_COR".

IMPORTANTE: Cada función se acompaña con

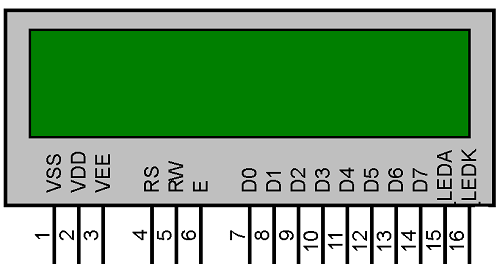
"INSTRUCCIÓN (NUM) <= FUNCIÓN ()”.



Esquemático del Funcionamiento de los módulos Procesador y Caracteres especiales



PINES DE LA LCD 16X2



La siguiente parte del módulo fue asignar los puertos de entradas y salidas digitalmente teniendo en cuenta que para este segundo módulo seguiremos haciendo uso de la FPGA Nexys 2.

Siguiendo la guía las referencias de esta FPGA son las siguientes:

Family: Spartan 3E, Device: XC3S500C, Package: FG320.

Con la herramienta Xilinx PlanAhead 14.7 se le asigno los pines de entrada y salidas digitalmente a nuestro diseño haciendo uso de la biblioteca proporcionada por INTESC.

NET "CLK" LOC = B8;

NET "DATA\_LCD[7]" LOC = L15;

NET "DATA\_LCD[6]" LOC = K12;

NET "DATA\_LCD[5]" LOC = L17;

NET "DATA\_LCD[4]" LOC = M15;

NET "DATA\_LCD[3]" LOC = K13;

NET "DATA\_LCD[2]" LOC = L16;

NET "DATA\_LCD[1]" LOC = M13;

NET "DATA\_LCD[0]" LOC = R18;

NET "ENA" LOC = R15;

NET "RS" LOC = T17;

NET "RW" LOC = P17;

NET "CORI" LOC = D18;

NET "CORD" LOC = B18;

Después de asignar los pines exitosamente se procedió a generar un archivo programable y con la Tarjeta desarrolladora a realizar las pruebas correspondientes conectar cada uno de los puertos asignados físicamente y por ultimo impactar la FPGA para evaluar que la información si se desplegué por la LCD 16X2 lo cual garantizaba que el diseño se había realizado con éxito.

Para este módulo se encontraron las siguientes dificultades:

* El desconocimiento del protocolo HD44780U (LCD-II)
* El desconocimiento de la biblioteca de INTESC la cual maneja los 8 pines de la LCD 16X2
* La asignación de pines en la FPGA digitalmente
* Las dificultades a la hora de conectar los puertos a la tarjeta desarrolladora físicamente.

# MODULO 3: DISPOSITIVO COLECTOR

Se inició una toma de decisiones por parte del equipo de trabajo del laboratorio para definir la cantidad total de sensores que utilizaremos en este módulo como el laboratorio dispone de una total de 10 a 12 sensores utilizaremos esta cantidad para realizar nuestro proyecto e implementar un nuevo módulo el cual definimos como COLECTO.VHD.

Antes de iniciar el desarrollo de este módulo colector se resaltan modificaciones importantes al módulo anterior (Módulo 2: Despliegue sobre pantalla LCD) ya que este módulo colector depende del módulo mencionado.

la siguiente modificación fue la asignación de pines de la FPGA nexys 2 que incluye el módulo 2 y los sensores sin una lógica de salida estos sensores siempre estarán en un NIVEL ALTO esto quiere decir que el sensor mientras no esté interrumpido siempre tendrá el valor lógico de 1 y si se interrumpe el sensor su valor lógico será 0, esta es su siguiente asignación:

NET "CLK" LOC = B8;

NET "RS" LOC = T17;

NET "CORD" LOC = B18;

NET "CORI" LOC = D18;

NET "RW" LOC = R15;

NET "ENA" LOC = R18;

NET "DATA\_LCD[0]" LOC = M13;

NET "DATA\_LCD[1]" LOC = U18;

NET "DATA\_LCD[2]" LOC = T18;

NET "DATA\_LCD[3]" LOC = R16;

NET "DATA\_LCD[4]" LOC = P17;

NET "DATA\_LCD[5]" LOC = L17;

NET "DATA\_LCD[6]" LOC = K12;

NET "DATA\_LCD[7]" LOC = L15;

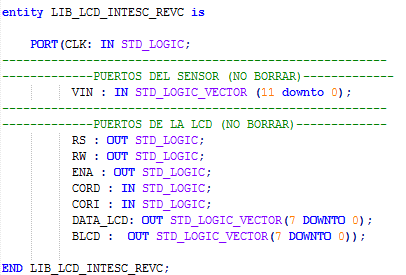
NET "VIN[0]" LOC = K13;

Se decidió por esta asignación ya que se puede conectar los pines de la LCD 16X2 de una forma más exacta y ordenada lo que nos va ayudar a optimizar el tiempo en las siguientes prácticas, este módulo puede utilizar los 12 sensores, pero solo se asignamos un pin de entrada del sensor a la FPGA para realizar la sustentación del módulo 3

Ya teniendo claro las modificaciones de los anteriores módulos procedemos

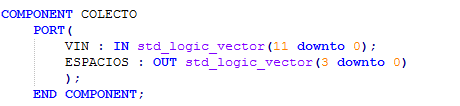
A diseñar el modulo colector:

Lo primero fue implementar las nuevas entradas de los sensores para esto utilizamos puerto VIN de tipo entrada lógica, siendo un vector de 12 posiciones (bits) la cantidad total de sensores que utilizaremos.

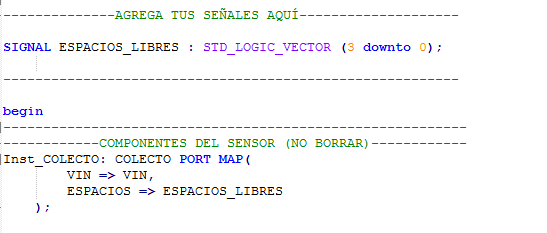


La instanciación del módulo del sensor y del módulo del LCD 16X2 en un solo proyecto.

instanciamos en el módulo LIB\_LCD\_INTESEC\_REV (Modulo Principal o Top Module) el modulo colector (COLECTO.VHD) copiando su componente y su instanciación, este módulo tiene una entrada VIN de tipo lógico y es un vector de 12 posiciones (Bits) y una salida ESPACIOS de tipo lógico también es un vector de 4 posiciones (Bits).

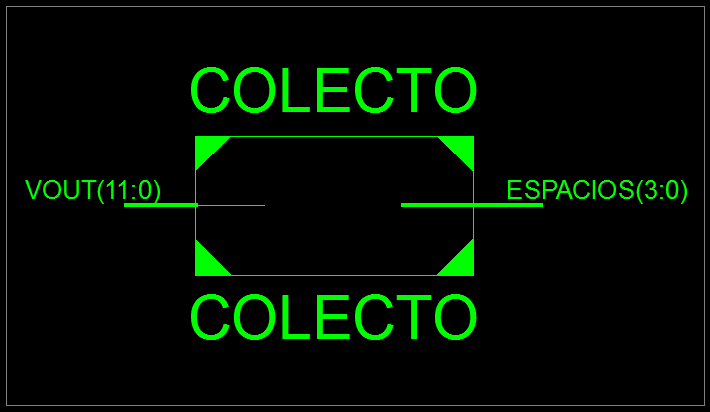


En el módulo LIB\_LCD\_INTESEC\_REV (Modulo Principal o Top Module) se crea una señal de tipo lógica ESPACIOS\_LIBRES siendo un vector de 4 posiciones (Bits) esta señal se le asigna a la salida de modulo COLECTO.VHD con esta señal desplegaremos la cantidad total de espacios libres en la LCD.



La entrada VIN del módulo LIB\_LCD\_INTESEC\_REV (Modulo Principal o Top Module) se le asigna a la entrada del módulo colector que tiene el mismo nombre VIN como las dos entradas son de tipo lógico y son un vector de 12 posiciones aseguramos la asignación de bit a bit.

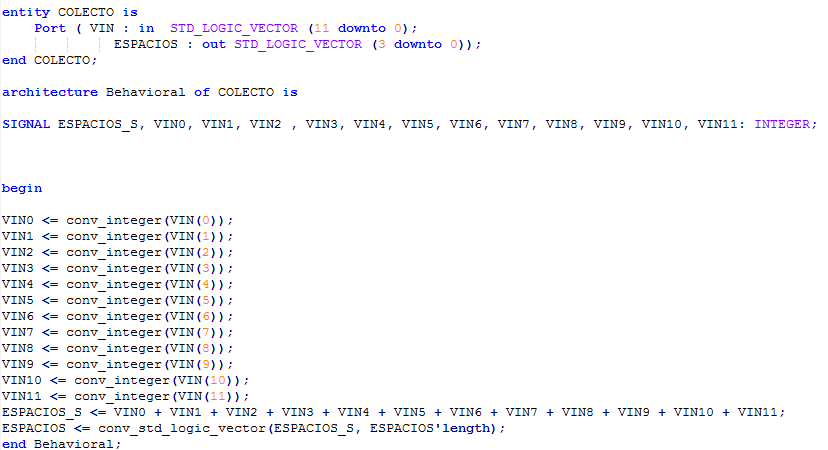
RTL Schematic COLECTO.VHD



Lo que sucede dentro dela caja negra básicamente consiste en que con la entrada VIN que contienen los valores de los sensores dependiendo de su estado generamos unas señales de tipo entero ESPACIOS\_S, VIN0, VIN2, …, VIN11

En cada señal de VIN# asignamos la conversión de las entradas de tipo lógico a entero luego de hacer esto le asignamos a la señal ESPACIOS\_S la suma de las señales VIN# por ultimo le asignamos a la salida ESPACIOS el valor de la señal ESPACIOS\_S de tipo entero, para esto tuvimos que hacer una investigación de como pasar de un entero a un vector de 4 posiciones(bits) de tipo lógico para esto se utiliza el comando:

conv\_std\_logic\_vector (signal, length);



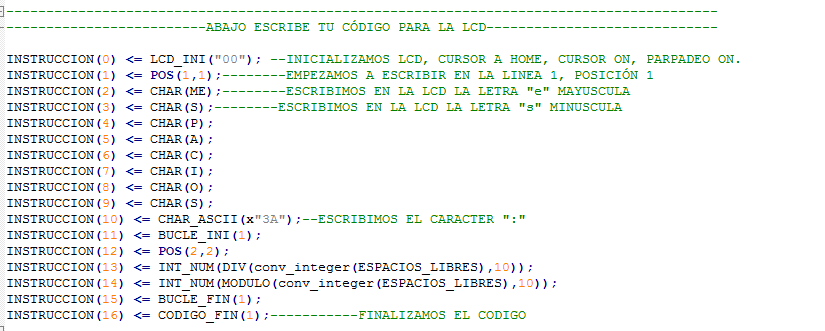
Que nos garantiza pasarle las señales directas de entrada de cada sensor a este módulo COLECTO.VHD que está siempre van a estar en un NIVEL ALTO, por lo tanto, la suma siempre va a estar en 12 (cantidad total de espacios disponibles) hasta que cambien los estados de un sensor (si detecta o no detecta) lo que hará que un sensor no sume con 1 si no con 0 reduciendo así los espacios disponibles.

Ya teniendo nuestro modulo colector pasamos al módulo LIB\_LCD\_INTESEC\_REV (Modulo Principal o Top Module) en este efectuaremos el despliegue de información en la LCD, para que la LCD este cambiando la información de la cantidad de espacios disponibles es necesario meter la INSTRUCCIÓN (12, 13 y 14) dentro de un bucle para eso utilizamos las funciones BUCLE\_INI () Y BUCLE\_FIN () Proporcionadas por la librearía que manejamos INTESEC con esto garantizamos que despliegue de información será continuo si se modifica el estado de los sensores.

Como ESPACIOS\_LIBRES es una señal lógica siendo un vector de 4 posiciones (bits) tenemos que convertir el valor binario que tenga esta señal a un entero para eso hacemos uso del siguiente comando:

conv\_integer (signal);

después lo pasamos a la LCD como un entero desplegando en pantalla la cantidad de espacios libres.



El siguiente mensaje a desplegar en la LCD seria:

Espacios:

# $

# ---> entero que indica las decenas

$ ---> entero que indica las unidades

Para hacer las divisiones y la operación modulo usamos una librería que nos permite hacer estas operaciones, esta Liberia también es diseña por la compañía Intescmx.

Se instancia de la siguiente forma:

* USE WORK.OP\_DIVISION.ALL

Se utiliza como se ve en la siguiente imagen:



Recibe 2 enteros separados por coma (“,”) dependiendo si es división DIV (A, B)

Y si es módulo MOUDLO (A, B) siendo A y B 2 enteros

Como la biblioteca de la LCD 16X2 solo nos deja mostrar enteros desde el 0 – 9

Se utilizan estas operaciones para desplegar la cantidad de espacios separando decenas y unidades por eso se utilizó esta biblioteca ya que algunas FPGA´s

No permiten la división (“/”) con esta biblioteca resolvemos este inconveniente

Para este módulo se encontraron las siguientes dificultades y soluciones:

* El desconocimiento de conversión de diferentes tipos de variables en VHDL

se inició una investigación obteniendo la información buscada.

* La asignación de pines en la FPGA digitalmente se asignó de manera más eficiente.
* Las dificultades a la hora de conectar los puertos a la tarjeta desarrolladora físicamente para esto se realizó una nueva asignación de pines.
* Los limitantes del hardware proporcionado para realizar esta práctica.

# MODULO 4: PROTOCOLO

Para la solución de este módulo se empezó una investigación por parte del equipo de trabajo del laboratorio para definir que protocolo de comunicación es más efectivo, la tarjeta de desarrollo a utilizar fue la nexys 2 la cual nos permite utilizar

Se descartó protocolo de Ethernet pues la FPGA carece de los puertos que soporta este mismo, entonces se decidió hacer uso del protocolo RS232 (UART) para la conexión entre las dos FPGA´s utilizando un puerto de transmisión y uno de recepción.

Antes de iniciar el desarrollo de este protocolo se resaltan las modificaciones importantes al módulo.

La siguiente modificación fue la asignación de pines de la FPGA nexys 2 que incluye el módulo. Asignamos las entradas de los sensores, los interruptores (Switches) de FPGA pues estos cumplen la misma función que un sensor enviarnos un 1 o 0 lógico, otro cambio importante fue que dejamos de utilizar la librería de la LCD 16x2 de 8 pines y pasamos a utilizar la LCD 16x2 de 4 pines, lo que nos ahorra utilizar todos los pines de la LCD. Seguimos utilizando el modulo colector que nos hace la suma de espacios disponibles que es controlado por los switches de la FPGA

NET "CLK" LOC = B8;

NET "VIN[0]" LOC = G18;

NET "VIN[1]" LOC = H18;

NET "VIN[2]" LOC = K18;

NET "VIN[3]" LOC = K17;

NET "VIN[4]" LOC = L14;

NET "VIN[5]" LOC = L13;

NET "VIN[6]" LOC = N17;

NET "VIN[7]" LOC = R17;

NET "RS" LOC = L17;

NET "RW" LOC = K12;

NET "ENA" LOC = L15;

NET "DATA\_LCD[0]" LOC = M16;

NET "DATA\_LCD[1]" LOC = M14;

NET "DATA\_LCD[2]" LOC = L16;

NET "DATA\_LCD[3]" LOC = K13;

Se decidió hacer esta asignación ya que se puede conectar los pines de la LCD 16X2 de 4 pines una forma más exacta y ordenada lo que nos va ayudar a optimizar el tiempo en las siguientes prácticas, este módulo puede utilizar los 8 switches y el resultado se ve reflejado en el módulo colector.

Ya teniendo claro las modificaciones de los anteriores módulos procedemos

A diseñar el protocolo RS232.

Se optó por la decisión de que el modulo central recibiera la información de otros (2 módulos colectores), cada módulo colector recibe a su vez 12 sensores.

El protocolo ya ha sido implementado por la empresa INTESC en una librería, para implementarla solo tuvimos que estudiar que funciones nos ayudaban a completar satisfactoriamente la transmisión y recepción entre las tarjetas de desarrollo.

Esta librería funciona de la siguiente manera:

Descripción: Las características de este módulo son:

\* 1 bit de inicio

\* 8 Bits de transmisión/recepción

\* 1 bit de Paro

\* Sin paridad

La recepción es asíncrona y podría ocurrir en cualquier momento.

Para saber cuándo se ha recibido un BYTE se usa RX\_IN. Este puerto se pone a '1' durante un ciclo de reloj cuando se ha recibido un byte.

Es responsabilidad del diseñador monitorearlo constantemente. El BYTE recibido estará disponible en DOUT. Por ejemplo, si se recibe un 55 hexadecimal, entonces RX, RX\_IN y DOUT se comportan de la siguiente manera

RX

¯¯¯¯¯¯\\_\_\_\_/¯¯¯¯\\_\_\_\_/¯¯¯¯\\_\_\_\_/¯¯¯¯\\_\_\_\_/¯¯¯¯\\_\_

RX\_IN

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/¯¯\\_\_\_\_

DOUT

--------------------------DESCONOCIDO--------><55------

La transmisión es controlada por 3 puertos: TX\_IN, TX\_FIN y TX.

Cuando se quiere comenzar a enviar, se debe poner TX\_INI a '1' y se debe cargar la información en DATAIN. Cuando se han terminado de enviar los 10 bits (1 bit de inicio, 8 bits de información y 1 bit de paro) entonces se pone TX\_FIN a '1' y no se permite otra transmisión hasta que TX\_INI sea '0'. EL diseñador debe poner TX\_INI y DATAIN. TX\_FIN se pone a '1' o a '0' automáticamente.

Por ejemplo, si se desea enviar el 55 hexadecimal y preparar para un nuevo envío, entonces la secuencia es la siguiente

TX

¯¯¯¯¯¯\\_\_\_\_/¯¯¯¯\\_\_\_\_/¯¯¯¯\\_\_\_\_/¯¯¯¯\\_\_\_\_/¯¯¯¯\\_\_

TX\_INI

\_\_\_\_\_/¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

TX\_FIN\_

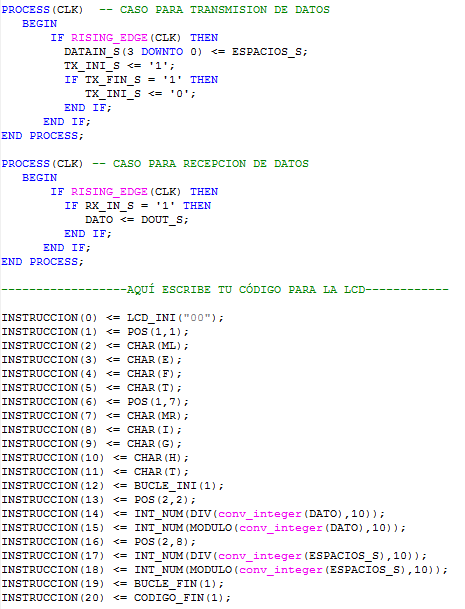
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/¯¯¯\\_\_\_\_

DATAIN

--------------------------55----------------------------- ><-------

No es necesario que DATAIN tenga la misma información durante toda la transmisión. Un registro interno copia la información cuando se inicia la transmisión dando la posibilidad de cambiar la información de DATA\_IN antes de que termine la transmisión sin problemas de colisión.

El código que se diseñó en el módulo de alto nivel es el siguiente:



El siguiente mensaje a desplegar en la LCD seria:

Left Rigth

#$ #$

# ---> entero que indica las decenas

$ ---> entero que indica las unidades

Para este módulo se encontraron las siguientes dificultades y soluciones:

* El desconocimiento de la implementación del protocolo RS232, con respecto a esta situación se inició una investigación obteniendo la información buscada.
* La asignación de pines en la FPGA digitalmente se asignó de manera más eficiente.
* Las dificultades a la hora de conectar los puertos a la tarjeta desarrolladora físicamente para esto se realizó una nueva asignación de pines.
* Los limitantes del hardware proporcionado para realizar esta práctica.

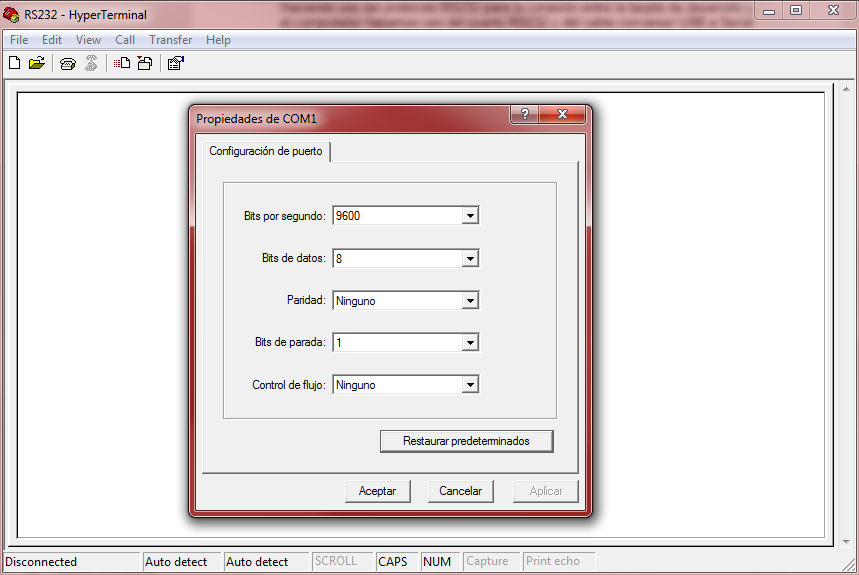
# MODULO 5: MODULO CENTRAL

Se inició una búsqueda de información por parte del equipo, que consta de una librería que maneje un puerto serial, el modulo en VHDL enviara la información desde la tarjeta de desarrollo nexys 2 donde se encuentra el colector principal a un DTE es este caso un computador que tenga un software terminal (Hyperterminal) Se hace uso del UART para enviar los datos de los dispositivos colectores DTE Haciendo uso del protocolo RS232 para la conexión entre la tarjeta de desarrollo y el computador hacemos uso del puerto RS232 y del cable conversor USB a Serial RS232.

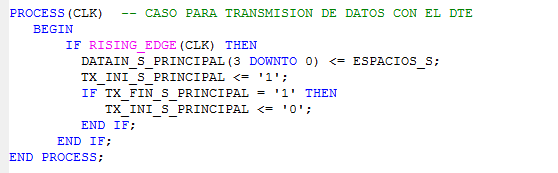


Para hacer uso de este cable se necesitan instalar unos drivers de tipo serial, lo que se hizo básicamente fue configurar el hyperterminal de acuerdo a los parámetros del protocolo que se está utilizando del RS232

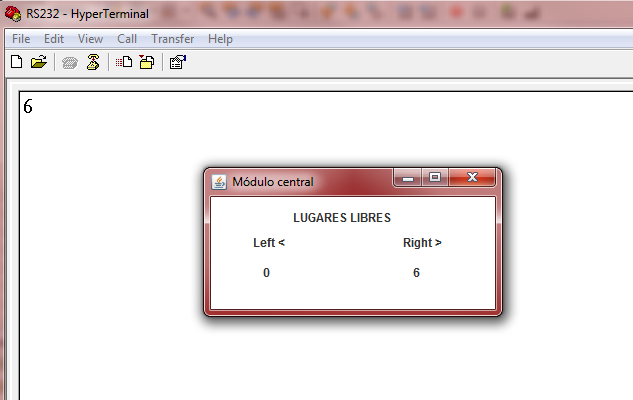
Para este caso sería una frecuencia de 9600 Bps un bit de inicio 8 bits de trasmisión/recepción y un bit de parada sin bit de paridad.



Después de haber configurado el hyperterminal preparamos el modulo para enviarlos datos desde los dispositivos colectores al hyperterminal



Con esto aseguramos él envió de datos desde la tarjeta de desarrollo al DTE Para realizar el software que nos despliegue la información de los dispositivos colectores se decide por utilizar como lenguaje de programación java haciendo uso de diferentes bibliotecas tales como la RXTX o la Giovynet que son librería muy conocidas en java, de desarrolla el software y se implementa una interfaz web sencilla para leer los datos recibidos desde el puerto serial en la siguiente imagen se muestra como el software recibe los datos y los plasma en la interfaz grafica



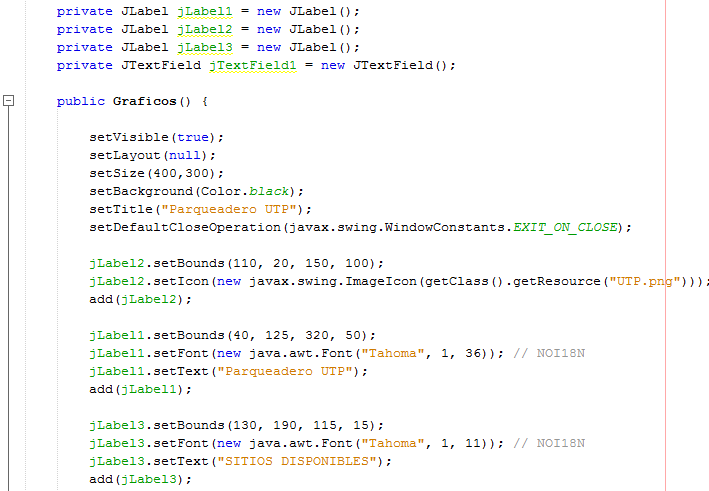
Para este módulo se encontraron las siguientes dificultades y soluciones:

* El desconocimiento de la implementación de la librería para leer los puertos
* La asignación de pines en la FPGA digitalmente se asignó de manera más eficiente.
* Los limitantes del hardware proporcionado para realizar esta práctica.

# MODULO 6: INTEGRACION DE LA SOLUCION

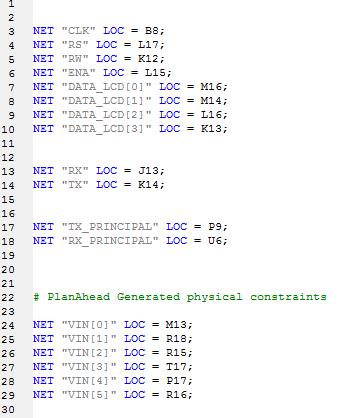
Módulo 6: Integración de la solución. Diseños adicionales y depuración de los desarrollos previos

Para la integración final de la solución, se desarrolló la siguiente interfaz gráfica utilizando java como lenguaje de programación predeterminado





Para este último modulo se depuro los errores tenían tanto los archivos. VHDL como los archivos .JAVA se optó por un último pineado para completar el proyecto este .UCF utiliza la LCD 16X2 de 4 bits y pineamos los sensores de los módulos colectores



Se desarrolló la siguiente maqueta para implementar lo hecho en laboratorio de electrónica digital, la cual nos servirá para dar la terminación del proyecto en todas sus condiciones



En la siguiente parte encontraremos la terminación del proyecto con los módulos 6 la integración de la solución.

Para este módulo se encontraron las siguientes dificultades y soluciones:

* El diseño de la interfaz Gráfica en Java.
* La asignación de pines en la FPGA digitalmente se asignó de manera más eficiente.
* El desarrollo de la maqueta para esta la terminación del proyecto.
* Los limitantes del hardware proporcionado para realizar esta práctica

# BIBLIOGRAFIA

## Márgenes para la presentación IEEE de un paper

Al crear un texto con la presentación IEEE, el primer paso que se debe hacer es configurar las márgenes, ya que esto determinara la apariencia del documento y afectara el espacio ocupado por los párrafos. Los principales aspectos a tener en cuenta son el tipo de papel, las márgenes y la división de la hoja.

1. <https://www.dynamoelectronics.com/>
2. <https://www.dynamoelectronics.com/hecho-en-colombia/433-sensor-digital-deteccion-5cm-con-board.html>
3. <https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-2/nexys2_rm.pdf>
4. <http://itatienredes.blogspot.com.co/2013/10/que-es-y-como-se-estructura-una-bitacora_8.html>
5. <https://www.sparkfun.com/datasheets/LCD/HD44780.pdf>
6. <https://www.youtube.com/watch?v=A7g4IkbV8PM&t=230s>
7. <http://www.intesc.mx/librerias/>
8. <https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-2/nexys2_rm.pdf>
9. <http://itatienredes.blogspot.com.co/2013/10/que-es-y-como-se-estructura-una-bitacora_8.html>
10. <https://www.nandland.com/vhdl/tips/tip-convert-numeric-std-logic-vector-to-integer.html>
11. <https://www.sparkfun.com/datasheets/LCD/HD44780.pdf>
12. <https://www.youtube.com/watch?v=A7g4IkbV8PM&t=230s>
13. <http://www.intesc.mx/librerias/>
14. <https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-2/nexys2_rm.pdf>
15. <https://www.youtube.com/watch?v=vR9WBsYrXtA>
16. <http://www.intesc.mx/librerias/>
17. <https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-2/nexys2_rm.pdf>
18. <https://www.youtube.com/watch?v=JAwCa4Uk6tA>
19. <http://www.intesc.mx/librerias/>
20. <http://fizzed.com/oss/rxtx-for-java>
21. <https://www.youtube.com/watch?v=jkP86HZxe5c>
22. <http://www.intesc.mx/librerias/>