Trabajo práctico final

Realizar un programa en lenguaje C que permita leer y analizar la información contenida en un archivo binario proporcionado por la cátedra. Dicha información se encuentra estructurada como se muestra a continuación y representa los paquetes de información que se quieren enviar desde un dispositivo (el emisor) hacia otros dispositivos (los receptores). Cada paquete se encuentra organizado de acuerdo al siguiente formato:

H1	**H2**	**ID**	**ORIG**	**DEST**	**TYPE**	**SIZE**	**DATA**	**CRC1**	**CRC2**
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	N bytes	1 byte	1 byte

H1 y H2: indican el inicio de un paquete de información, cuyos valores constantes son H1= **0x3C** y H2 = **0x4D**. Estos valores deben comprobarse para determinar el inicio de un nuevo paquete de información.

ID: es el identificador del paquete. Dado que se trata de un canal de comunicación asincrónico, el orden en que llegan los paquete NO necesariamente es el mismo en el que fueron generados. ID corresponde al índice en que se generaron.

ORIG: es el identificador del dispositivo que genera información, (**0x01** en este caso).

DEST: es el identificador del dispositivo al que se destina la información. Puede tomar dos valores: **0x05** o **0x10** (Dos posibles destinos en este caso).

TYPE: es el tipo de información, puede adoptar los valores **0x30** o **0x40**. Esta información la utiliza sólo el equipo transmisor.

SIZE: es el tamaño (en bytes) del campo **DATA**.

DATA: corresponde a una secuencia de **SIZE** bytes que representa los números enteros (en el rango 0 a 4096) enviados a los diferentes dispositivos.

CRC1 y **CRC2** corresponden al cálculo de un CRC de 16 bits. El CRC es un código de detección de errores usado frecuentemente en redes digitales y en dispositivos de almacenamiento para detectar cambios accidentales en los datos. La función que implementa este cálculo es

suministrada por la cátedra mediante 2 archivos: uno de cabecera, “crc_16.h”, y el correspondiente código compilado “crc_16.o”. Estos 2 archivos deben copiarse en el directorio donde se encuentra el proyecto y el header debe incorporarse al mismo. El archivo *.o si bien se encuentra en el directorio del proyecto, NO debe incorporarse al mismo. Además también debe copiarse en el directorio \obj\Debug\. Para chequear la integridad del paquete se debe llamar a la función “calculate_checksum”, cuya firma se muestra a continuación:

uint16_t calculate_checksum(uint8_t[], uint16_t);

Si el valor de retorno de esta función es CERO significa que el paquete llegó sin modificaciones ni errores. Un valor distinto de CERO indica que se produjo un error y el paquete debe descartarse.

El programa deberá contemplar la recepción de 3 parámetros de entrada que corresponden al identificador del equipo destino (0x05 o 0x10), el nombre del archivo de entrada y el nombre del archivo de salida.

Una vez leídos todos los datos según las observaciones anteriores:

A. se deben ordenar los paquetes correlativamente de acuerdo a su campo ID.

B. los valores enteros ordenados deben normalizarse en el rango [0,1] con la precisión de un tipo double.

C. Se deben guardar en un archivo de texto de acuerdo al siguiente formato:

```
línea 1            x   ;   y    \n
línea 2          orden0; valor0 \n
línea 3          orden1; valor1 \n
                       .
línea N+1              .
```

D. Abrir el archivo de texto generado con el archivo salida.xlsm a partir del gráfico que se obtiene especifique el tipo de onda, su valor pico a pico, su valor medio y su período.

El programa será probado desde la línea de comandos ingresando el nombre del archivo.exe id_destino archivo_entrada

archivo_salida. Deben validarse TODOS los parámetros de entrada al main().