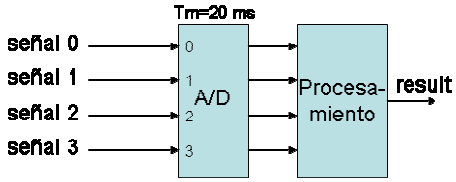
1) *3 puntos*

Se desea realizar un programa para la adquisición y procesamiento de un conjunto de señales analógicas, según el esquema siguiente:



Las especificaciones son las siguientes:

- La función main() solicitará al usuario el valor ***n*** a utilizar en el cálculo del resultado (ver más adelante). No se puede suponer a priori un límite máximo para ***n***.
- A continuación, main() llamará a la función:

  void IniciarAD(int Tm, float* valores, int nvalores,void (*FnProcesa)(float* datos, int ndatos));

  que cada Tm milisegundos lee los canales A/D, guarda los valores leídos en la tabla indicada, y llama a la función deseada por el programador para el procesamiento de estos valores (a la cual pasa la tabla y el nº de datos leídos).
- Todo el procesamiento se realiza en la función callback a desarrollar, la cual debe realizar el cálculo siguiente y escribir el resultado en pantalla:

  result = Media de los últimos 5 valores de la señal AD_0 +

  Producto Escalar últimos ***n*** valores señal AD_1 x últimos ***n*** valores señal AD_2 +

  Acumulado señal AD_3.

2) *2.5 puntos*

Realizar un programa que realice un bucle que solicite por consola una cadena de caracteres, busque **dentro de la cadena** la aparición de un texto con el formato:

```
... ENVIAR <<direccion_ip>> <<puerto>> <<texto>> ...
```

y realice el envío del texto solicitado, mediante socket con conexión (TCP), a la dirección IP y puerto indicados (ver funciones en documentación anexa).

Entre cada campo del formato anterior puede haber un número indeterminado de espacios.

3) *2.5 puntos*

Un sistema de T.R. dispone de 3 tareas que deben realizar sus cálculos en función del estado de las entradas de un puerto de E/S digital de 32 bits.

- La tarea 1 se activa bajo interrupción, y debe añadir a un archivo en modo texto el valor de 32 bits en formato hexadecimal (cadena de formato "%x").
- La tarea 2 debe comprobar cada 30 segundos el estado del bit de peso 2. Si está activo, debe poner a 1 el bit de peso 5 y lanzar a la tarea 3 pasándole como parámetro un valor entero entre 0 y 10, que se incrementa cada vez que esto ocurre (si pasa 10, vuelve a 0). Si el bit de peso 2 sigue activo 5 segundos después, debe poner a 0 el bit de peso 5 y detener a la tarea 3.
- La tarea 3 debe generar en el bit de peso 6 del mismo puerto una onda PWM, de periodo 10 segundos, y tiempo duty el valor entero pasado como parámetro.
- Escribir un programa que ejecute estas 3 tareas con los mecanismos de sincronización adecuados (ver funciones en documentación anexa), disponiendo de una librería de acceso al puerto de E/S digital con los siguientes archivos:

C:\LibreriaES\include\es.h	C:\LibreriaES\lib\es.lib
int InicializaES(int *ptid,void (*RutServInt)(void)); /* Inicializa dispositivo de E/S y establece la rutina de servicio de interrupción. Devuelve -1 si hay error o 0 si no hay error, y en el 2º caso en *ptid el identificador de dispositivo */ int GetES(int id); /* Lee el estado de los bits del puerto de 32 bits del dispositivo indicado por 'id' */ void PutES(int id, int valor); /* Escribe el estado de los bits del puerto de 32 bits del dispositivo indicado por 'id' */	…

- Indicar cómo haría en Visual C++ para que el programa pueda ser compilado y enlazado con esta librería + otras librerías necesarias.
- Se encuentra en el archivo el dato 0037B8A6. ¿Cuál era el valor de los bits 2, 5 y 6?

4) *1 punto*

Cuestiones (no más de **2** líneas por cuestión):

a) ¿Qué es una excepción?

b) ¿Qué problemas puede generar en una aplicación de T.R.?

c) ¿Qué mecanismo POSIX y en qué forma (síncrona/asíncrona) se utiliza para manejar una excepción?

d) ¿Cuál es la diferencia entre el tratamiento de una excepción y el de cualquier otra señal?

5) *1 punto*

Cuestiones (no más de **4** líneas por cuestión):

a) ¿Por qué no se pueden utilizar las funciones de E/S de propósito general (fopen, fread, fscanf, …) para la adquisición de datos en T.R.?

b) Ponga un ejemplo para la obtención de un dato entero pasado como parámetro de ejecución de un programa.

Paso de parámetros a main() desde línea de comandos:

```
int main(int argc,char* argv[])  // argv[1] es una cadena de caracteres para el
                                 // 1er parámetro, argv[2] para el 2º, etc.
```

Algunas funciones de cadenas de caracteres:

```
int atoi(const char* cadena);
int strlen(const char* cadena);
char* strcpy(char* dst,const char* src);
char* strncpy(char* dst,const char* src,int n);
char* strcat(char* dst,const char* src);
char* strncat(char* dst,const char* src,int n);
char* strchr(const char* cad,char c);
char* strstr(const char* cad,const char* busca);
```

Documentación de sockets:

```
SOCKET sock;

SOCKET socket(int address_family,int sock_type,SOCK_STREAM,int protocol);
// Usar address_family=AF_INET, sock_type = SOCK_DGRAM o SOCK_STREAM, protocol=0

int bind(SOCKET sock,struct sockaddr_in *addr_local,int tam_sockaddr_in);
int connect(SOCKET sock, struct sockaddr_in *addr_remoto,int tam_sockaddr_in);
int listen(SOCKET sock,int maximo);
int accept(SOCKET sock, struct sockaddr_in *addr_remoto,int tam_sockaddr_in);
int send(SOCKET sock,char* datos,int nbytes,int flags); // Usar flags=0
int recv(SOCKET sock,char* datos,int nbytes,int flags); // Usar flags=0
int closesocket(SOCKET sock);


struct sockaddr_in {
        short   sin_family;
        u_short sin_port;
        struct  in_addr sin_addr;
        char    sin_zero[8];
};

Para dirección IP, usar:
       addr_laquesea.sin_addr.S_un.S_addr=inet_addr(cad. caract. dirección ip);

Para el Puerto, usar:
       addr_laquesea.sin_port=htons(entero nº de puerto);
```

Documentación de funciones y estructuras POSIX utilizables:

```
int pthread_create(pthread_t* id,pthread_attr* attr,void *(*fn)(void*),void* arg);
                    // arg es un puntero al parámetro que se quiere pasar al hilo
                    // usar attr=NULL

int pthread_cancel(pthread_t thread);

int sem_init(sem_t *sem, int pshared, unsigned int value); // Usar pshared=0
int sem_wait(sem_t* sem);
int sem_post(sem_t* sem);
int sem_destroy(sem_t* sem);

int pthread_mutex_init(pthread_mutex_t* mutex, const pthread_mutexattr_t *attr);
                                        // Usar attr=NULL
int pthread_mutex_lock(pthread_mutex_t* mutex);
int pthread_mutex_unlock(pthread_mutex_t* mutex);
int pthread_mutex_destroy(pthread_mutex_t* mutex);

int pthread_cond_init(pthread_cond_t* cond, const pthread_condattr_t *attr);
                                        // Usar attr=NULL
int pthread_cond_wait(pthread_cond_t* cond,pthread_mutex_t* mutex);
int pthread_cond_broadcast(pthread_cond_t* cond);
int pthread_cond_destroy(pthread_cond_t* cond);

int sigemptyset(sigset_t* set);
int sigaddset(sigset_t* set,int signo);
int pthread_sigmask(int how, const sigset_t *set, sigset_t *oset);
                    /*     how=SIG_BLOCK, SIG_SETMASK ó SIG_UNBLOCK */
int sigaction(int sig, const struct sigaction *act,struct sigaction *oact);
int sigwait(sigset_t* set,int* sig);

struct sigaction
{
      void (*sa_sigaction)(int signo, siginfo_t *info, void *dontuse);
      sigset_t sa_mask;
      int sa_flags; /* Poner bit SA_SIGINFO */
};

int timer_create(clockid_t clockid,struct sigevent* sig,timer_t* id);
                                  // Usar clockid= CLOCK_MONOTONIC y sig=NULL
int timer_settime(timer_t id,int flags,const struct itimerspec* ival,
                  struct itimerspec* oval);   // Usar flags=0, oval=NULL
int timer_delete(timer_t id);

struct timespec
{
      time_t tv_sec;      // Segundos
      long tv_nsec;// Nanosegundos
} ;

struct itimerspec
{
      struct timespec it_interval;     // Periodo de temporización
      struct timespec it_value;        // Primera activación
} ;

mdq_t mq_open(char* nombre,int apert,mode_t modo,mq_attr* attr);
      // 'apert' puede ser combinación con | de: O_CREAT, O_WRONLY, O_RDONLY. Si
      'apert' contiene O_CREAT, hay que añadir los parámetros modo=0 y attr, sino
      no se ponen.

mdq_t mq_open(char* nombre,int apert,mode_t modo,mq_attr* attr);
int mq_send(mdq_t idcola,char* datos,int nbytes,int prioridad);
int mq_receive(mdq_t idcola,char* datos,int nbytes,int* prioridad);
int mq_unlink(mdq_t idcola);

typedef struct mq_attr {
      long mq_flags; /*usar 0 */
      long mq_maxmsg; /*número máximo de mensajes*/
      long mq_msgsize; /*tamaño máximo del mensaje*/
      long mq_curmsgs; /*actual número de mensajes*/
} mq_attr_t;
```