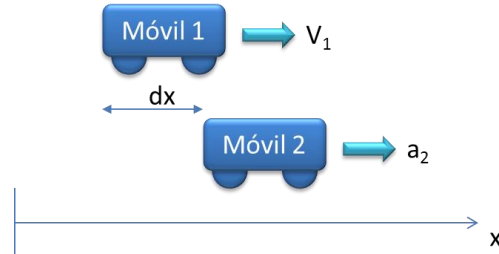


# Control de Procesos en Tiempo Real

## Examen Extraordinario - Julio 2013

5 puntos 1)

Escribir un programa que lea valores de velocidad/aceleración de dos móviles en los canales A/D 1 y 2 cada **50 ms**, y escriba por pantalla la diferencia de las posiciones **en metros** de ambos móviles ( $dx$ ).



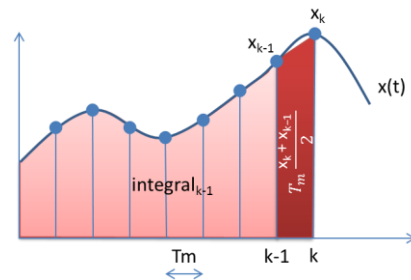
Las condiciones de funcionamiento son las siguientes:

- Se dispone de las siguientes funciones para la conversión A/D:

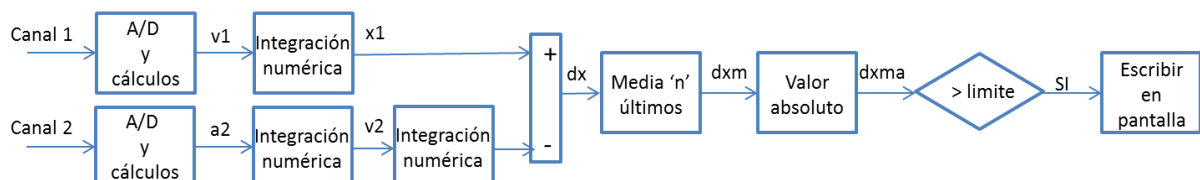
```
// Inicialización de un temporizador que llame a la función indicada en
// FnCallback cada cierto intervalo (en ms)
void InitTemporizador(int tiempo_ms,void (*FnCallback)());

// Obtención del valor de conversión A/D del canal deseado utilizando
// nBits : resultado de 0 a 2^n-1, para tensión de entrada 0 a 5V.
int ConversionAD(int nCanal,int n);
```

- Se desea utilizar conversiones A/D de 10 bits, con conversores de entrada 0-5V.
- La medida de velocidad del canal 1 se obtiene a partir de un tacómetro que da 1V para  $v=-50$  Km/h, y 4V para  $v=50$  Km/h.
- La medida de aceleración del canal 2 se obtiene a partir de la suma de 2.5V a la tensión generada por un acelerómetro de 500 mV/g ( $g=9.81$  m/s<sup>2</sup>).
- La integración numérica se realiza mediante la aproximación trapezoidal:



- Al comienzo del programa principal, se solicita valor entero que indica el tiempo en ms ( $T_{media\_ms}$ ) que se debe considerar para calcular la diferencia de posición.
- El programa principal se queda en un bucle, solicitando un valor real, que es el límite máximo de la diferencia de posición admitida (en metros).
- Si, en algún instante, el valor medio de las  $n$  últimas diferencias de posición de ambos móviles (actual y recientes dentro del tiempo  $T_{media\_ms}$ ) es mayor en valor absoluto que el límite máximo, se debe indicar mediante un mensaje en pantalla.



# Control de Procesos en Tiempo Real

## Examen Extraordinario - Julio 2013

---

2) 2 puntos

En el siguiente programa, el hilo principal solicita los valores  $t1max$  y  $t2max$ , y el hilo secundario comprueba si dos variables  $t1$  y  $t2$  son mayores que  $t1max$  y  $t2max$  para generar una alarma. Realizar las modificaciones necesarias para:

```
void* Hilo2(void* p)
{
    ...
    while (1)
    {
        t1=LeerAD(...);
        t2=LeerAD(...);
        if (t1>t1max && t2>t2max)
            ;;;; ALARMA !!!!
    }
}

main()
{
    ...
    while (1)
    {
        ...
        scanf("%f", &t1max);
        scanf("%f", &t2max);
        ...
    }
}
```

- Declarar en el lugar y con el tipo adecuado las variables  $t1$ ,  $t2$ ,  $t1max$  y  $t2max$ . Justificar el porqué de cada decisión.
- Escribir el código necesario en el lugar adecuado para lanzar el hilo 2.
- Hacer que el acceso a las variables compartidas esté correctamente protegido.
- Despertar a un 3<sup>er</sup> hilo cuando se cumple la condición de alarma.

3) 2 puntos

Escribir una **función** que, dados como parámetros el nombre de un archivo de texto (que contiene líneas de texto), la extensión de dicho archivo, y una cadena a buscar en dicho archivo, devuelva el nº de líneas del archivo que contienen ese texto.

Condiciones:

- No se puede suponer a priori el tamaño máximo del nombre y la extensión de archivo, o de la cadena de texto.
- Se debe leer el archivo línea por línea (función `fgets`).
- Ninguna línea del archivo supera 256 caracteres.
- Ejemplo de llamada a la función:

```
main()
{
    int n;

    n=MiFuncion("datos", "txt", "es");
    // n debe valer 2
}
```

datos.txt

```
En esta línea aparece el texto buscado.
esta línea tiene dos veces el texto, sólo cuenta una.
Esta línea no tiene el texto buscado.
Y ésta tampoco.
```

4) 1 punto

Contestar **con una sola palabra** a cada una de las siguientes preguntas:

- a) ¿Qué mecanismo se utiliza para tratar las excepciones en POSIX?
- b) Si un programa se bloquea a la espera de una operación de E/S, tiene un funcionamiento .....
- c) La conexión de un programa a la red se caracteriza en cada extremo por la dirección IP y el .....
- d) La variable descriptora de una conexión de red se denomina .....

# Control de Procesos en Tiempo Real

## Examen Extraordinario - Julio 2013

---

### 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 C:

```
int atoi(const char* cadena);
double atof(const char* cadena);
double strtod(const char* cadena, char** final);

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);
int strcmp(const char* cad1, const char* cad2);
int strncmp(const char* cad1, const char* cad2, int n);
char* strchr(const char* cad, char c);
char* strstr(const char* cad, const char* busca);
int sprintf(char* dest, const char* fmt, ...);
int sscanf(const char* src, const char* fmt, ...);

FILE* fopen(const char* filename, const char* mode);
char* fgets(char* dest, int tam_max, FILE* f);
int fprintf(FILE* fid, const char* fmt, ...);
int fscanf(FILE* fid, const char* fmt, ...);
int fclose(FILE* fid);

void* malloc(int tam);
void free(void* pt);
```

### 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, const 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 n° de puerto);
```

# Control de Procesos en Tiempo Real

## Examen Extraordinario - Julio 2013

---

### Documentación de funciones y estructuras POSIX utilizables:

```
int pthread_create(pthread_t* id, pthread_attr* attr, void *(*fn)(void*), void* arg);
int pthread_cancel(pthread_t thread);
void* pthread_join(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);
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);
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
};

mqd_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. */

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