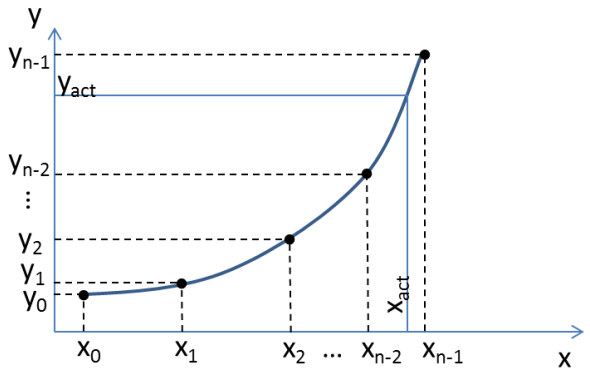
Grado en Ingeniería Electrónica y Automática
Informática Industrial y Comunicaciones

Examen Ordinario – Enero 2020

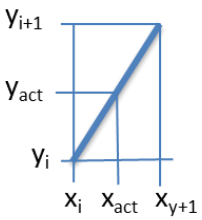
1) *1.5 puntos*

Realizar una función que, dados como parámetros dos tablas (xi, yi) con los valores (crecientes) que expresan la relación no-lineal entre las variables *x* e *y*, y un valor real y_act, obtenga el valor x_act correspondiente más aproximado. La función devolverá -1 si el valor y_act es menor que y_0 o mayor que y_{n-1}.



Algoritmo sugerido:

- Buscar en la tabla yi el 1er valor i tal que:
  $y_i < y\_act <= y_{i+1}$
- Realizar la interpolación lineal entre los valores i e i+1 para obtener x_act:



2) *1.5 puntos*

Realizar una función que, dado como parámetro el nombre de un archivo de texto que contiene 10 líneas con valores reales x y según el formato adjunto, rellene dos tablas de reales con los contenidos de dicho archivo.

datos.txt

```
0.5 V   -> 5.3  %
1   V   -> 15.25%
2.75V   -> 27.8 %
...
```

En el archivo pueden existir espacios en blanco entre los valores numéricos y las unidades, y a ambos lados de la flecha (->). No es necesario comprobar errores de formato del archivo.
El resultado de la función para los datos de ejemplo debe ser:
Tabla x a la salida: 0.5, 1.0, 2.5, …
Tabla y a la salida: 5.0, 15.2, 27.8, …

3) *1.5 puntos*

Realizar una función que, dados como parámetros los siguientes valores:

- Un entero estado (**modificable** por la función) cuyos bits se deben interpretar como:

Bits →	…	B5	B4	B3	B2	B1	B0
Interpretación →	?	Auto activado	?	?	Freno activado	?	?

- Un valor real acel_k con el valor actual del acelerador en %.
- Una tabla de 4 valores reales errv_k (**no modificable** por la función) con el valor actual y anteriores de error de velocidad en Km/h
- Una tabla de 8 valores reales aper_k (**no modificable** por la función) con el valor actual y anteriores de apertura de válvula de gasolina en %

realice el siguiente algoritmo:

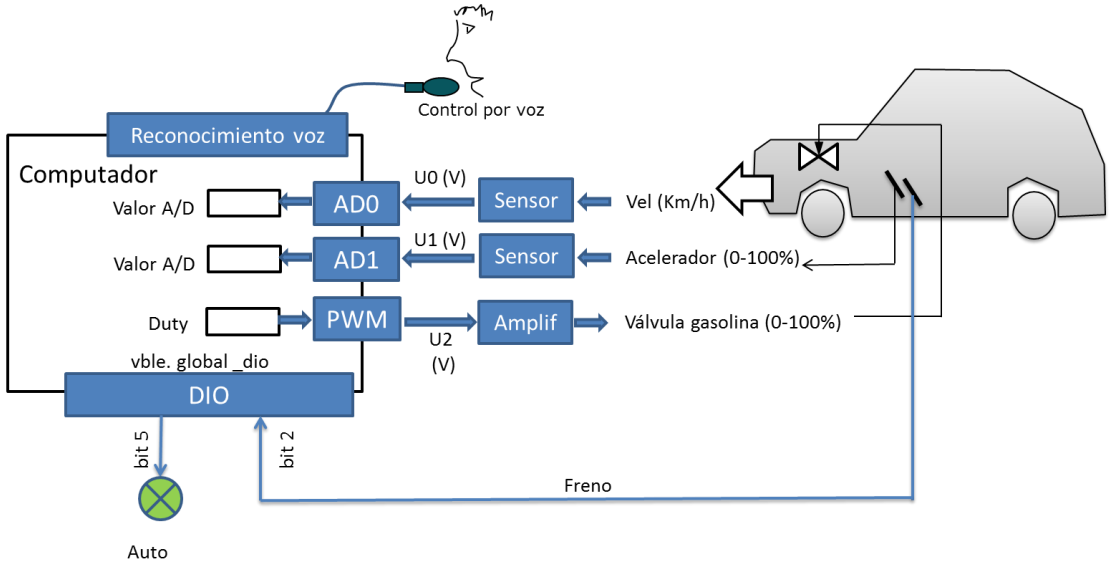
Grado en Ingeniería Electrónica y Automática
Informática Industrial y Comunicaciones

Examen Ordinario – Enero 2020

- Ponga a 0 el bit B5 de **estado** si el freno está activado o el valor actual **acel_k[0]** es menor que 0.
- Devuelva el valor actual **acel_k[0]** si el bit B5 de estado está a 0 (modo manual).
- Devuelva el resultado de llamar a la función `ControlAuto()` (ya existente en librería FnAux, ver anexo) si el bit B5 está a 1 (modo automático).

4) Se desea realizar un sistema de control de velocidad automático de un automóvil.

4 puntos

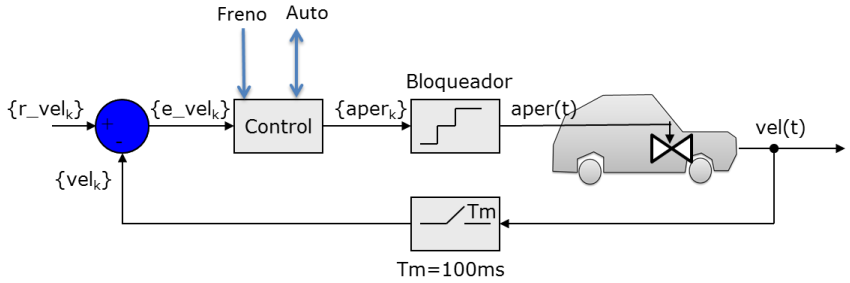


- La velocidad actual (**vel_k**) es captada por un sensor que entrega una tensión entre 0.6V (para 0Km/h) y 4.5V (para 180Km/h), y adquirida mediante un conversor A/D (canal 0) de 12 bits.
- La presión sobre el acelerador (**acel_k**) es captada por un sensor no lineal y adquirida mediante un conversor A/D (canal 1) de 10 bits. Para obtener la presión a partir de la tensión leída del conversor se utilizará la función del ejercicio 1 con los datos de la curva leídos de un archivo de texto "datos.txt" según la función del ejercicio 2.
- La apertura de la válvula de gasolina (**aper_k**) se acciona mediante una señal PWM (0-100%).
- El conductor puede introducir mediante su voz los comandos siguientes:
  - **AUTO**: activa el seguimiento automático de la velocidad (bit **auto** a 1), y pone como referencia de velocidad (**rvel_k**) la velocidad actual del vehículo.
  - **MANUAL**: desactiva el seguimiento automático de la velocidad (bit **auto** a 0).
  - **AUMENTAR**: incrementa en un 5% o 5Km/h (el valor más grande de ambos) la velocidad de referencia actual (**rvel_k**).
  - **REDUCIR**: decrementa en un 5% o 5Km/h (el valor más grande de ambos) la velocidad de referencia actual (**rvel_k**).

  Un módulo de reconocimiento de voz entregará una cadena de caracteres con el texto expresado por el conductor.
- El control a realizar se resume en la figura siguiente:

Grado en Ingeniería Electrónica y Automática
Informática Industrial y Comunicaciones

Examen Ordinario – Enero 2020



Control() se ejecutará cada 100 ms por interrupción, con el siguiente algoritmo:

- Si freno activado o acelerador pisado, pasar a modo manual (apagar LED modo auto).
- Si modo manual, $aper_k = acel_k$
- Si modo auto, se utiliza la ecuación en diferencias:

$$aper_k = b_0.e\_vel_k + b_1.e\_vel_{k-1} + \ldots + b_3.e\_vel_{k-3} - (a_1.aper_{k-1} + a_2.aper_{k-2} + \ldots + a_7.aper_{k-7})$$

- El programa principal realiza las inicializaciones necesarias, lee del archivo "datos.txt" la tabla para la interpretación del sensor de aceleración (ejercicio 2), establece modo manual y lanza una interrupción temporizada cada 100 ms. A continuación, realiza un bucle infinito en que espera con la función `ReconocimientoDeVoz()` (similar a gets( ) y existente en librería FnAux, ver anexo) la llegada de una línea de texto pronunciada por el usuario. Si se reconoce la orden, se actúa sobre las variables correspondientes para ejecutarla según la especificación.

- En la interrupción temporizada se realizarán, por este orden:
  - El desplazamiento de las tablas de valores temporales utilizadas.
  - Obtener la velocidad actual del vehículo a partir del dato leído en el canal AD 0, calcular el error actual respecto a la referencia, y añadirlo a la tabla de errores.
  - Obtener la presión actual del conductor sobre el acelerador (0-100%) a partir del dato leído en el canal AD 1 y la función del ejercicio 1.
  - Llamar a la función del ejercicio 3 para recalcular el valor manual/auto (valor digital en variable global _dio) y obtener el valor de apertura de válvula a aplicar. Añadir este valor en su tabla temporal.
  - Aplicar el valor de apertura de válvula actual mediante la función `AplicarPWM()` (ya existente en librería FnAux, ver anexo).

Grado en Ingeniería Electrónica y Automática
Informática Industrial y Comunicaciones

Examen Ordinario – Enero 2020

ANEXO

Librería auxiliar:

Se dispone de las siguientes funciones dentro de una librería FnAux (archivos FnAux.h y FnAux.lib):

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

// Aplicación de la señal PWM correspondiente a una apertura deseada de
// la válvula de gasolina:
void AplicarPWM (float pct_apertura_0_a_100);

// Obtención del texto pronunciado por el usuario mediante reconocimiento de
// voz. Funciona similar a gets():
:
void ReconocimientoDeVoz(char* destino);

// Cálculo de la apertura de la válvula de gasolina (0% a 100%) en caso de
// control automático:
float ControlAuto( const float errvel_k[],int nvel_k,
                   const float apert_k[],int n_apertk);
```

Algunas funciones de C:

```
int atoi(const char* cad);              // Devuelve entero equivalente a cadena
double atof(const char* cad);           // Devuelve real equivalente a cadena
double strtod(const char* cad,char** next);   // Id. a atof() y guarda en next puntero
                                              // a final de conversión
int strlen(const char* cadena);         // Devuelve longitud de cadena
char* strcpy(char* dst,const char* src);      // Copia cadena fuente en destino
char* strncpy(char* dst,const char* src,int n);      // Id. Máximo n caracteres
char* strcat(char* dst,const char* src);      // Concatena cadena Fuente a destino
char* strncat(char* dst,const char* src,int n);      // Id. Máximo n caracteres
char* strchr(const char* cad,char c);  // Busca caracter en cadena, devuelve puntero
                                       // a la primera ocurrencia o NULL si no está
char* strstr(const char* cad,const char* busca);     // Id. buscando cadena
int strcmp(const char* c1,const char* c2);    // Compara cadenas, devuelve 0 si iguales
char* gets(char* destino);              // Lee cadena de consola, almacena en destino
void* malloc(int n_bytes);              // Asigna memoria para n bytes
void free(void* ptr);                   // Libera memoria asignada
FILE* fopen(const char* nombre,const char* modo);   // Abre stream
char* fgets(char* dest,int n_max,FILE* fid); // Lee línea de stream de texto
int fscanf(FILE* fid,...); // Lee datos de stream de texto con formato
int fprintf(FILE* fid,...); // Escribre datos en stream de texto con formato
void fclose(FILE* fid); // Cierra stream
```

Grado en Ingeniería Electrónica y Automática
Informática Industrial y Comunicaciones

Examen Ordinario – Enero 2020

APELLIDOS Y NOMBRE: _____

Ejercicio 4 (responder aquí):
1.5 puntos

a) ¿Cuál de las siguientes afirmaciones sobre una comunicación serie RS-232 es **falsa**? Marcar la respuesta apropiada.

- ☐ Se puede realizar una comunicación RS-232 únicamente con 3 hilos: emisión, recepción y masa común.
- ☐ RS-232 necesita un archivo de configuración llamado "/dev/ttyS0" (en sistemas Linux).
- ☐ Los dos extremos de una comunicación RS-232 deben usar la misma velocidad de reloj (baudrate).
- ☐ Se programa una comunicación RS-232 siguiendo un esquema similar a la escritura/lectura de un archivo.

b) Ordenar (1 a 4) la secuencia de llamadas a funciones de socket para realizar un cliente TCP:

Orden	Instrucción
	bind(sock,&dir_ip_local,sizeof(struct sockaddr_in));
	send(sock,data,n_bytes,0);
	sock=socket(AF_INET,SOCK_STREAM,0);
	connect(sock,&dir_ip_remoto,&tam_ip_remoto);

c) ¿Cuánto vale la variable *x* tras ejecutar el código siguiente?

```
void Suma(const int* t,int n,float result);   /* Suma los valores de una tabla
...                                            de n enteros */
int datos[4]={0,2,4,6};
float x ;
Suma(datos,4,x);
```

- ☐ *x* vale 12 si la función Suma( ) está correctamente realizada
- ☐ No se puede compilar, el 1er parámetro de Suma( ) no puede ser const
- ☐ No se puede compilar, la tabla debería ser de float
- ☐ No se puede saber el valor de *x*, independientemente de lo que haga Suma( )