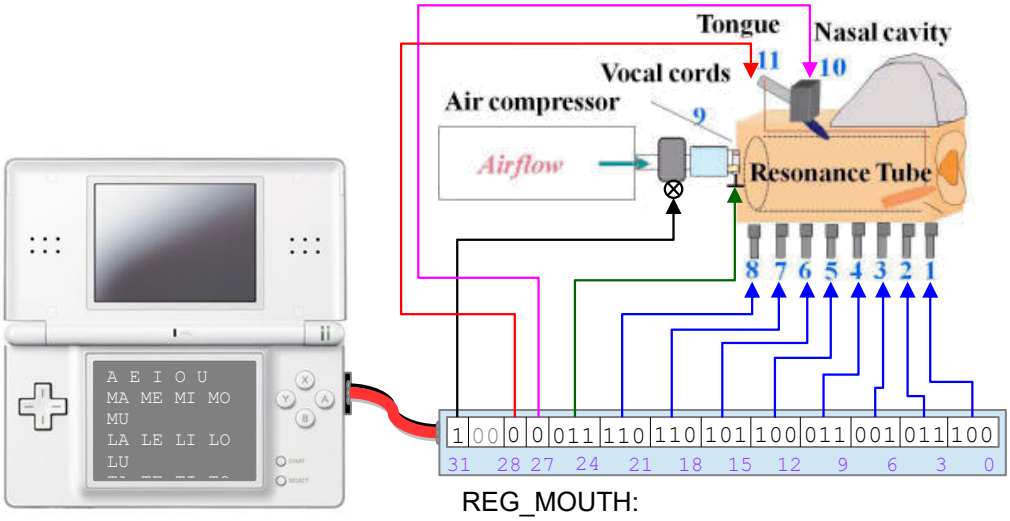
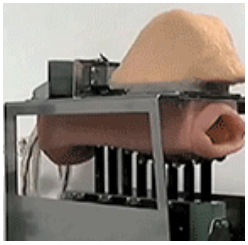
Problema 24: Boca artificial (Ex. 1ª Conv. 2017-18)

Hay que controlar una boca artificial que permite simular la vocalización humana (hasta cierto punto). Este dispositivo consiste básicamente en un tubo de resonancia por el que circula un flujo de aire que proviene de un deposito de aire comprimido. El tubo es flexible y su forma se puede modificar con una serie de 8 pistones que pueden generar 8 elevaciones diferentes cada uno. También se pueden regular 8 niveles de apertura de una válvula que simula las cuerdas vocales, una lengua artificial que se puede subir o bajar, y una cavidad nasal a la cual se puede desviar la salida del aire para reproducir sonidos nasales. Por último, hay una electroválvula que permite abrir o cerrar el flujo general del aire. A continuación se muestra un esquema de la conexión entre la NDS y el dispositivo, además de una foto real de dicho dispositivo:



Artificial Mouth:



El registro de Entrada/Salida REG_MOUTH permite controlar todos los actuadores del dispositivo mediante los siguientes campos:

Campo	*Bits*	*Descripción*
`Tube shape`	23..0	En grupos de 3 bits, indican el nivel de elevación de los 8 pistones que dan forma al tubo de resonancia, donde los 3 bits de menor peso controlan el pistón 1, los siguientes 3 bits controlan el pistón 2, etc.; el nivel 000 corresponde a la elevación más baja y el nivel 111 a la más alta
`Vocal cords`	26..24	Indican el nivel de apertura de la válvula que simula las cuerdas vocales, donde el nivel 000 es el más cerrado y el nivel 111 es el más abierto

`Nasal cavity`	27	Indica si el aire tiene que entrar en la cavidad nasal artificial (1) o no (0)
`Tongue`	28	Indica si la lengua artificial tiene que estar arriba (1) o abajo (0)
`Air flow`	31	Indica si se abre (1) o cierra (0) el flujo general de aire

El propósito general del programa a realizar es recibir las palabras a vocalizar a través de la wifi de la NDS, transcribir dichas palabras a "fonemas" específicos para el dispositivo propuesto (posición de los actuadores), visualizar la palabra actual en la pantalla inferior de la NDS y enviar cada fonema al registro de E/S durante el tiempo asociado al fonema, todo ello mientras se realizan una serie de tareas independientes.

Se dispone de las siguientes rutinas, ya implementadas:

Rutina	*Descripción*
`inicializaciones()`	Inicializa el *hardware* (pantalla, interrupciones, etc.)
`tareas_independientes()`	Tareas que no dependen de la vocalización de las palabras; supondremos que nunca tardarán más de 100 milisegundos en ejecutarse
`activar_timer0(int frec)`	Activa el funcionamiento del *timer* 0 a la frecuencia indicada por parámetro (en Hz), incluida la activación de las interrupciones correspondientes (IRQ_TIMER0).
`desactivar_timer0()`	Desactiva el funcionamiento del *timer* 0
`int pal2fon(char *pal, unsigned int fon[], unsigned char tim[])`	Transcribe una palabra que se pasa por parámetro como *string* (`pal`) a una secuencia de valores con la posición de los actuadores del dispositivo (excepto bit 31) sobre un vector que se pasa por referencia (`fon`), junto con el tiempo (en centésimas) de cada fonema sobre otro vector pasado por referencia (`tim`), y devuelve como resultado el número total de fonemas generados; el tiempo mínimo de ejecución de esta rutina es de 10 milisegundos
`swiWaitForVBlank()`	Espera hasta el próximo retroceso vertical
`printf(char *format,...)`	Escribe un mensaje en la pantalla inferior de la NDS

Se propone usar las siguientes variables globales y definiciones:

```
#define MAX_CAR     16
#define MAX_FON     50
char palabra[MAX_CAR+1];           // palabra a vocalizar
unsigned int fonemas[MAX_FON];     // fonemas a vocalizar
unsigned char fcs[MAX_FON];        // tiempo de cada fonema, en
                                   // centésimas de segundo
unsigned char num_fon = 0;         // número de fonemas a vocalizar
```

El vector `palabra[]` permitirá almacenar los caracteres de la palabra actual a vocalizar, suponiendo que nunca recibiremos palabras de más de 16 caracteres. Los vectores `fonemas[]` y `fcs[]` permitirán almacenar el valor de los actuadores y el tiempo en centésimas de los fonemas correspondientes a la palabra actual. Por último, la variable `num_fon` permitirá almacenar el número total de fonemas obtenidos, suponiendo que nunca habrá más de 50 fonemas para cualquier palabra a vocalizar.

Para gestionar el envío de fonemas al registro de E/S, se pide usar la RSI del *timer* 0 para detectar si se ha agotado el tiempo del fonema actual y, en caso afirmativo, que active el siguiente fonema, si es que hay fonemas pendientes. En caso de haber terminado con la vocalización del último fonema, habrá que cerrar la electroválvula del paso general del aire y desactivar el *timer* 0.

El inicio de la vocalización del primer fonema se tiene que gestionar desde el programa principal, además de activar el *timer* 0 cuando se reciba una nueva palabra a vocalizar.

Para detectar la recepción de una nueva palabra, se pide realizar una rutina específica:

```
int siguiente_palabra(char *string);
```

la cual recibe por parámetro la dirección inicial de un vector sobre el cual almacenará los caracteres de la palabra, además de añadir un carácter centinela de final de *string* ('\0'). Como resultado hay que devolver la longitud del *string* (sin contar el centinela), o cero si no se ha recibido ninguna palabra en el momento de la llamada a la rutina.

Debido a que la comunicación con la wifi solo la puede gestionar el procesador ARM7 y el programa a realizar se ejecutará íntegramente sobre el ARM9, se pide utilizar el mecanismo IPC-FIFO de la NDS para traspasar el texto a vocalizar, palabra a palabra, como secuencias de caracteres sobre la cola FIFO de recepción del ARM9. Hay que recordar que esta cola dispone de 16 posiciones (*words*), de modo que el ARM7 podrá copiar una palabra entera

sobre la cola, enviando hasta 16 caracteres seguidos cuando reciba una notificación de que la cola está vacía (IRQ_FIFO_EMPTY).

Por su parte, la rutina `siguiente_palabra()` podrá detectar si hay alguna palabra pendiente en la cola consultando el bit 8 del registro `REG_IPC_FIFO_CR`, que valdrá 1 si la cola de recepción está vacía o 0 si no lo está. Para obtener todos los caracteres almacenados en la cola, habrá que ir leyendo el registro `REG_IPC_FIFO_RX` (32 bits) secuencialmente, hasta que la cola esté vacía.

Se pide:

Programa principal en C, RSI del *timer* 0 y rutina `siguiente_palabra()` en ensamblador.