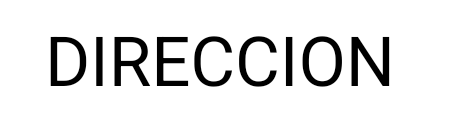
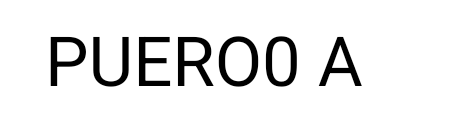
**Problema 1**

Dado el siguiente mapa de registros



Tabla

El contenido generado por IA puede ser incorrecto.Tabla

El contenido generado por IA puede ser incorrecto.

Realizar las siguientes funciones

Propósito: Establecer un pin de puerto como entrada o salida;

Recibe como parámetros número de puerto, numero de pin y dirección (0 entrada, 1 salida)

Retorna 0 por exito o -1 por error (número de puerto mayor que tres, número de pin mayor que 31, dirección diferente de 0 o uno)

**int SetDir( unsigned char port , unsigned char pin, unsigned char direccion);**

Propósito: Colocar un cero o un uno en un pin de un puerto;

Recibe como parámetros número de puerto y número de pin

Retorna 0 por éxito o -1 por error (número de puerto mayor que tres o número de pin mayor que 31)

**int SetPin( unsigned char port , unsigned char pin);**

Propósito: Leer el valor de un pin de un puerto;

Recibe como parámetros número de puerto y número de pin

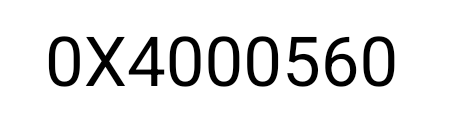
Retorna valor leído por éxito o -1 por error (número de puerto mayor que tres o número de pin mayor que 31)

**int GetPin( unsigned char port , unsigned char pin);**

Nota: Considere que los registros ubicados a partir de la dirección 0xA0003000 son aquellos que nos permiten acceder a los puertos mientras que los ubicados a partir de 0xE0003000 modifican su modo de operación.

**Problema 2**

Dado el siguiente mapa de registros.



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

|  | Canal 0 a 7 |
| --- | --- |
|  | Valor 0 a 4096 |

Realizar una función que reciba un puntero a estructura y cargue en sus campos el numero de canal y el valor medido.

En donde la estructura deberá tener el siguiente formato

struct canales

{

unsigned char canal;

unsigned int valor;

};

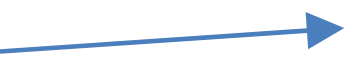
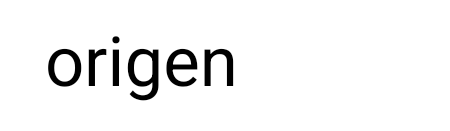
void LecturaDeCanal( struct canales \* cn );

**Problema 3**

Realizar una función que reciba un puntero a una strings cargada con ceros y unos (siempre 8 valores) y un puntero a un unsigned char tal que la convierta a un número entero y la cargue en lo apuntado por el puntero a unsigned char que recibió;

**void convertir (char \* origen , unsigned cha Destino );**

Ejemplo



| ‘0’ | ‘1’ | ‘0’ | ‘1’ | ‘0’ | ‘1’ | ‘0’ | ‘1’ | ‘\0’ |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |

Daría por resultado el numero 85

**Problema 4**

Se desea controlar el semáforo de una calle de una sola mano con las siguientes condiciones:

**Para los autos:**

* Luz verde durante 15 segundos.
* Luz amarilla durante 2 segundos.
* Luz roja durante 10 segundos.

**Para los peatones:**

* Luz peatonal verde (para cruzar) sólo cuando el semáforo de autos está en rojo.
* Luz peatonal roja en el resto del ciclo

Además, existe un botón que, al ser presionado en cualquier momento, activa un modo especial:

* El semáforo de autos comienza a parpadear (enciende y apaga la luz amarilla cada segundo).
* Este modo de parpadeo se mantiene hasta que el botón sea presionado nuevamente, momento en el cual el semáforo vuelve a su funcionamiento normal.

Se pide:

* Realizar la máquina de estados
* Realizar la codificación de la función de la máquina de estados en C, mediante la estructura de control de flujo switch case
* Realizar la codificación de la función de la máquina de estados en C, mediante la estrategia de punteros a función.

Utilizar para la realización las siguientes funciones de biblioteca

#define ROJO 0

#define VERDE 1

#define AMARILLO 2

**Parámetros**:

color: color de la luz del semáforo de Autos

estado: ON o OFF

void LuzAutos( unsigned char color, unsigned char estado);

**Parámetros**:

color: color de la luz del semáforo Peatonal

estado: ON o OFF

void LuzPeatonal( unsigned char color, unsigned char estado);

**Parámetro**:

Retorna el valor leído: cero no pulsado uno pulsado.

unsigned char Pulsador( void );

**Proposito**: Dispara un temporizador

n: Numero de timer

t: Cantidad de tiempo

base: DEC,SEG,MIN

Evento Puntero a función de la función que se ejecuta cuando vence el temporizador

void TimerStart( char n , unsigned int t, char base , void (\*Evento)(void) );

**Propositp**: Apaga un timer

n: Numero de timer

void TimerStop( char n);

**Propósito**: Analiza el vencimiento de los temporizadores, se debe colocar en el lazo principal

void TimerEvent( void );