Apellidos: Nombre:

DNI:

1. El WS2812B es un diodo led RGB inteligente muy usado actualmente. El color del led se programa mediante un pin de entrada de comunicación serie utilizando un protocolo propietario. Además, dispone de un pin de salida de datos que le permite conectarse en serie a otro diodo led (y este a un tercero, y así sucesivamente). Gracias a esta característica, es fácil utilizar este led para construir tiras de ledes, anillos, matrices, etcétera.

   El protocolo de programación es muy sencillo. En reposo, la entrada se mantiene a nivel bajo, estado que cada led propaga también a su salida. La programación comienza cuando empiezan a enviarse los datos bit a bit, con un total de 24 bits por led (suponiendo una cadena de varios ledes). Cada led individual retiene los primeros 24 bits que le llegan (manteniendo mientras tanto su salida a nivel bajo) y deja pasar los siguientes. El proceso termina cuando la entrada de programación se vuelve a poner a nivel bajo durante al menos 50 µs. En este momento, cada led actualiza su color con los datos que ha recibido en el proceso de programación. Para que este protocolo pueda funcionar, se requiere de una codificación especial para los bits. Así, el valor 0 para un bit se codifica manteniendo la señal a nivel alto durante 0,40 µs y bajo durante 0,85 µs, y el valor 1 con la señal a nivel alto durante 0,80 µs y bajo durante 0,45 µs. De esta manera, mientras se envían bits de datos es imposible que la línea esté a nivel bajo durante el tiempo que indicaría el final de la programación.

   Supongamos que se tiene un anillo formado por 12 de estos ledes, conectados en serie. Se desea saber:

   a) La latencia con respecto al quinto led del anillo.

   b) La productividad máxima cuando se programan todos los ledes.

   c) La productividad necesaria para un efecto que requiera cambiar el color de todos los ledes 10 veces por segundo.

   (Puesto que en el examen no se puede utilizar calculadora, no es necesario que realices los cálculos, tan solo deja indicado cómo se tendrían que calcular los resultados que se piden.)

   Como indica el enunciado, para programar el color de uno de estos ledes es necesario enviar 24 bits (8 por componente RGB). Pese a que las codificaciones para un bit a 0 o a 1 son diferentes, se puede ver que la duración de ambos es la misma: 1,25 µs (0,40 + 0,85 µs para el 0 y 0,80 + 0,45 µs para el 1).

   Así pues, programar un led costará $24\,\text{bits} \times 1{,}25\,\mu\text{s/bit} = 30\,\mu\text{s}$ (sin tener en cuenta el tiempo de finalización de la programación). Veamos cómo responder a las preguntas partiendo de este dato.

   a) Para calcular la latencia del quinto led basta con tener en cuenta que desde que se inicia la operación de programación del anillo se suministran datos para los 4 ledes anteriores y sólo entonces comienzan a llegar los datos al quinto led. De esta manera, la latencia pedida es de $4\,\text{ledes} \times 24\,\text{bits/led} \times 1{,}25\,\mu\text{s/bit} = 120\,\mu\text{s}$.

b) Para calcular la productividad máxima para programar los 12 ledes del anillo, se podría entender o que no se tiene en cuenta el tiempo de finalización —aunque en tal caso no se programarían—, o que sí. En cualquiera de los dos casos, los datos a enviar son 12 ledes × 24 bits/led = 288 bits.

Si no se tiene en cuenta el tiempo de finalización, el tiempo sería el calculado para un led, por 12. Con esto, la productividad sería: $\frac{12\,\text{ledes}\times 24\,\text{bits/led}}{12\,\text{ledes}\times 24\,\text{bits/led}\times 1{,}25\,\mu\text{s/bit}} = 0{,}8\,\text{bits/}\mu\text{s} = 800.000\,\text{bits/s}$.

Si tenemos en cuenta el tiempo de finalización, que sería el mínimo, puesto que se pide la productividad máxima, habría que sumar 50 µs al denominador anterior, con lo que el resultado sería: $\frac{12\,\text{ledes}\times 24\,\text{bits/led}}{50\,\mu\text{s}+12\,\text{ledes}\times 24\,\text{bits/led}\times 1{,}25\,\mu\text{s/bit}} \approx 702.439\,\text{bits/s}$.

c) Para programar completamente los ledes 10 veces por segundo, y dado que el tiempo ya viene fijado —se pregunta la productividad necesaria—, solo hay que evaluar los datos a enviar en un segundo. Como cada led consume 24 bits y el anillo completo, 12 ledes × 24 bits/led, programarlos 10 veces requiere 12 ledes/vez × 24 bits/led × 10 veces/s, es decir, 2.880 bits/s. Dado que la productividad máxima es muy superior, es evidente que es posible realizar el efecto sugerido con este anillo.

2. La mayor parte de los relojes inteligentes incorporan aplicaciones deportivas o de salud que permiten medir el pulso cardíaco e incluso, algunas de ellas, representar gráficamente su ciclo. Para ello, estos relojes disponen de un dispositivo de E/S que: I) emite luz con una longitud de onda muy precisa, II) capta y mide la luz reflejada en las venas subcutáneas, y III) proporciona la intensidad de la señal reflejada, que el procesador del reloj utilizará para determinar la fase del ciclo cardíaco del usuario.

   Indica el comportamiento e interlocutor y da una estimación razonada de la productividad y de la latencia de un sensor de pulso como el descrito. Ten en cuenta que debe ser capaz de tomar las medidas necesarias para que se pueda representar el ciclo cardíaco y que el pulso humano puede variar entre 60 y 120 pulsaciones (ciclos) por minuto.

   

   (Puesto que en el examen no se puede utilizar calculadora, no es necesario que realices los cálculos, tan solo deja indicado cómo se tendrían que calcular los resultados que se piden.)

   El comportamiento del sensor es de entrada, dado que entrega al reloj —que como sabemos, es un computador completo con esta forma física— los datos correspondientes a la intensidad de la luz reflejada, convertidos a valores digitales.

   El interlocutor es, evidentemente, humano. Un ser humano lleva el reloj, calibrado para leer la reflexión de la luz emitida, en la sangre humana, a través de la piel humana.

   La productividad tiene que permitir tomar suficientes muestras en un ciclo cardíaco para obtener una gráfica. Si suponemos que cada valor de luz se convierte a 10 bits —una resolución razonable en conversores AD genéricos de calidad media—, se toman al menos 10 muestras por ciclo cardíaco, a una frecuencia máxima de 120 ciclos por minuto, se obtiene una productividad de:
   10 bits/muestra × 10 muestras/ciclo × 120 ciclos/minuto = 12.000 bits/minuto = 200 bits/s.

   Por último, la latencia tiene que ser baja, del orden de ms o menos, pues los ciclos cardíacos, como se ve en la imagen, se pueden representar en tiempo real dando una imagen fidedigna

del estado del usuario.

3. Ciertos microcontroladores, para facilitar su diseño, combinan en un mismo registro, bits de estado, de control e incluso de datos. Así pues, se puede hablar, en lugar de registros de control, estado o datos, de bits de control, estado o datos. Indica razonadamente, para los siguientes bits de dispositivos para la transmisión serie de información —lo que se conoce como USART —, cuáles serían de control, cuáles de estado y cuáles de datos.

Bits	Tipo
`SPEN`: *Serial Port Enable bit.* Sirve para habilitar el dispositivo USART.	de control
`RX9`: *9-bit Receive Enable bit.* Permite habilitar el tamaño de datos de 9 bits para la recepción.	de control
`RX9D`: *9th bit of Received Data.* Almacena el noveno bit del dato recibido.	de datos
`TRMT`: *Transmit Shift Register Status bit.* Indica si el registro de transmisión está lleno o vacío.	de estado
`OERR`: *Overrun Error bit.* Indica error de sobrescritura.	de estado

   Los bits `SPEN` y `RX9` son de control, pues permiten habilitar características del dispositivo —o habilitar el propio dispositivo en sí—, siendo por tanto configuraciones o activaciones que debe generar el procesador.

   El bit `RX9D` es un bit de datos, pues como indica la descripción contiene el 9º bit del dato recibido por el dispositivo durante la comunicación.

   Los bits `TRMT` y `OERR` son bits de estado. El primero avisa al procesador de una circunstancia normal durante el funcionamiento del dispositivo, como es que el registro de transmisión está vacío porque ya se ha enviado vía serie o leído el dato a enviar o recibir. El segundo informa de un error en la transmisión o recepción detectado por el dispositivo.

4. Explica el mecanismo de sincronización de la entrada/salida mediante consulta de estado e indica sus ventajas e inconvenientes.

   El mecanismo de consulta de estado consiste en que el procesador, durante la ejecución de los programas, lea cada cierto tiempo los registros de estado de los dispositivos que gestiona y verifique los bits que indican que requieren atención.

   Su principal ventaja es que es sencillo de implementar y no requiere de circuitería adicional más allá de la propia de un procesador estándar y un dispositivo simple de E/S. También se tiene que, en el caso de un sistema totalmente dedicado a gestionar un dispositivo de E/S, presenta la menor latencia. Si bien este no es el caso habitual.

   Sus inconvenientes tienen que ver con el tiempo, cuando el programa debe realizar otras tareas más allá de consultar la E/S. En este caso se tiene que el sistema pierde tiempo consultando bits de estado de dispositivos que no requieren atención; que la latencia es muy variable, y en general alta, teniendo en cuenta la relación temporal entre el momento en que el dispositivo activa el bit de estado y aquél en que el programa consulta este bit.

   Por último, está la dificultad de insertar las instrucciones de consulta de estado en un programa que además se tiene que dedicar a otras tareas.

5. Describe brevemente los dos modos de transferencia de datos, indicando cuándo es más adecuado cada uno de ellos.

   Cuando un sistema debe trasladar datos desde un dispositivo a la memoria principal, o viceversa, existen dos mecanismos:

   Por programa, cuando el procesador ejecuta instrucciones que leen los datos de una fuente (dispositivo o memoria) y los escriben en el destino (memoria o dispositivo, respectivamente).

   Mediante acceso directo a memoria (DMA). En este caso, el sistema incorpora un dispositivo especial, llamado controlador de DMA, que realiza las copias de los datos con independencia del procesador, que queda libre para realizar otras tareas.

   La transferencia por procesador es preferible cuando deban copiarse pocos bytes, con un tamaño similar al de los registros a programar en el controlador de DMA para realizar una transferencia por este método. Cuando se deben transferir bloques de datos de mayor tamaño, es recomendable el uso del DMA.