Examen final

(Duración: 3 horas)

Nombre: _____ **Grupo:** _____

INSTRUCCIONES

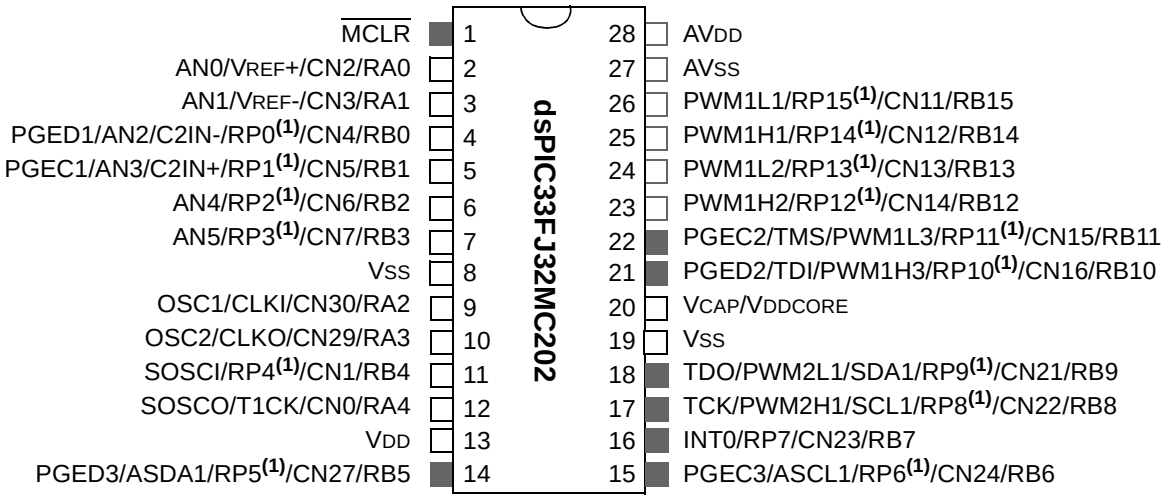
- No abra el cuadernillo hasta que se le indique.
- El examen consiste en un sistema electrónico que se ha dividido en **tres problemas** con un valor total de **diez puntos**.
- Se recomienda leer el enunciado completo antes de empezar a resolver el examen.
- Cada uno de los problemas deberá realizarse en una **hoja de solución diferente**.
- En la página 3 y siguientes encontrará un **formulario** que puede resultarle útil.
- Se permite el uso de **calculadora** y de una **hoja con anotaciones** sobre acondicionamiento de sensores.
- No desgrape el cuadernillo. Puede utilizar las últimas páginas como **borrador**.
- Si lo prefiere, puede resolver el examen a **lápiz**.
- Deberá emplear **máscaras** para leer o escribir los bits de los registros. La única excepción son los bits asociados a las interrupciones, con los que puede utilizar el acceso a nivel de bit.

Calificación:

FORMULARIO

Microcontrolador dsPIC33FJ32MC202

Pines



Timers[1]

- `TMRx`: Cuenta del timer
- `PRx`: Valor final de la cuenta
- `TxCON`: Registro de configuración
  - `TON`: Bit de encendido
    `1` = Enciende el timer
    `0` = Apaga el timer
  - `TCKPS`: Preescalado
    `3` = 1:256
    `2` = 1:64
    `1` = 1:8
    `0` = 1:1

15	14	13	12	11	10	9	8
TON		—					

7	6	5	4	3	2	1	0
	—	TCKPS			—	—	

- Función de atención a la interrupción

```
void __attribute__((interrupt, no_auto_psv)) _TxInterrupt(void);
```

[1] x indica el número de Timer (1, 2 ó 3).

Conversor analógico-digital

- `AD1PCFGL`: Registro para configurar los pines como analógicos o digitales
- `AD1CHS0`: Permite seleccionar el pin analógico que se va a convertir
- `ADC1BUF0`: Almacena el resultado de la última conversión realizada
- `AD1CON3`: Registro de configuración #3
  - `ADCS`: Preescalado
    - T_{AD} = Periodo del conversor A/D
    - `TCY`: Periodo del oscilador interno

$$T_{AD} = TCY \cdot (ADCS + 1)$$

  - `SAMC`: Número de ciclos de muestreo

15	14	13	12	11	10	9	8
–			SAMC				

7	6	5	4	3	2	1	0
ADCS							

- `AD1CON1`: Registro de configuración #1
  - `ADON`: Arranca el periférico pero no empieza a muestrear
    `1` = Enciende el módulo
    `0` = Apaga el módulo
  - `SSRC`: Indica cuándo comienza la conversión
    `7` = Automáticamente al terminar de muestrear
  - `SAMP`: Inicio del muestreo (se borra automáticamente)
    `1` = Empezar a muestrear

15	14	13	12	11	10	9	8
ADON		–			–	–	

7	6	5	4	3	2	1	0
SSRC				–	–	SAMP	–

- Función de atención a la interrupción

```
void __attribute__((interrupt, no_auto_psv)) _ADC1Interrupt(void);
```

Generador de PWM[2]

- `PxTPER`: Permite configurar el periodo de la señal (15 bits)

```
PxTPER = Periodo · FCY · preescalado − 1
```

- `PxDCy`: Factor de servicio de la señal del pin `PWMxHy` (16 bits)
  - Si es igual que `PxTPER` equivale a un 50 % de factor de servicio
- `PWMxCON1`: Registro de control de puertos
  - `PENyH`: Habilita el puerto `PWMxHy`
    `1` = Salida de PWM
    `0` = Entrada o salida digital
  - `PENyL`: Habilita el puerto `PWMxLy`
    `1` = Salida de PWM
    `0` = Entrada o salida digital

15	14	13	12	11	10	9	8
					–		

7	6	5	4	3	2	1	0
	PEN3H	PEN2H	PEN1H		PEN3L	PEN2L	PEN1L

- `PxTCON`: Registro de configuración
  - `PTEN`: Bit de encendido
    `1` = Enciende el módulo
    `0` = Apaga el módulo
  - `PTCKPS`: Preescalado para calcular la frecuencia de la señal
    `3` = 1:64
    `2` = 1:16
    `1` = 1:4
    `0` = 1:1

15	14	13	12	11	10	9	8
PTEN		–					

7	6	5	4	3	2	1	0
–				PTCKPS		–	

[2] x indica el número de generador de PWM (1 ó 2) e y el pin dentro de cada generador.

Registros de control de interrupciones

- `IFS0`: Banderas de interrupción

15	14	13	12	11	10	9	8
		AD1IF	–	–	–	–	T3IF

7	6	5	4	3	2	1	0
T2IF	–	–		T1IF	–	–	–

- `IEC0`: Registro para habilitar la atención de interrupciones

15	14	13	12	11	10	9	8
		AD1IE	–	–	–	–	T3IE

7	6	5	4	3	2	1	0
T2IE	–	–		T1IE	–	–	–

- `IPCx`: Prioridad de interrupciones
  - `IPC0`

15	14	13	12	11	10	9	8
	T1IP				–		

7	6	5	4	3	2	1	0
	–				–		

  - `IPC1`

15	14	13	12	11	10	9	8
	T2IP				–		

7	6	5	4	3	2	1	0
	–				–		

  - `IPC2`

15	14	13	12	11	10	9	8
	–				–		

7	6	5	4	3	2	1	0
	–					T3IP	

  - `IPC3`

15	14	13	12	11	10	9	8
					–		

7	6	5	4	3	2	1	0
	AD1IP				–		

En vista de que los estudios de mercado auguran un repunte de la venta de cámaras de fotos durante la campaña navideña, una empresa líder del sector ha contactado con usted para que diseñe la electrónica de un nuevo accesorio, un **trípode motorizado** compatible con todos sus modelos equipados con disparo por infrarrojo[3], que esperan lanzar a tiempo para las rebajas como parte de un *pack* especial.

Mecánicamente, el trípode no es más que una plataforma *pan-tilt*, es decir, que puede rotar en el plano horizontal XZ (en inglés, *pan*) y en el vertical YZ (*tilt*) como se aprecia en la Figura 1. Lo que se pretende es incorporar una serie de sensores y actuadores para que se puedan hacer fotos a distancia perfectamente horizontales. Además, como quieren demostrar la rapidez de su algoritmo de *image stitching*, recientemente patentado, el trípode contará con un modo de funcionamiento que hará girar la cámara 360º tomando instantáneas solapadas para construir una imagen panorámica. Para abaratar costes, en esta primera versión no se incluirá el ajuste del *roll*.

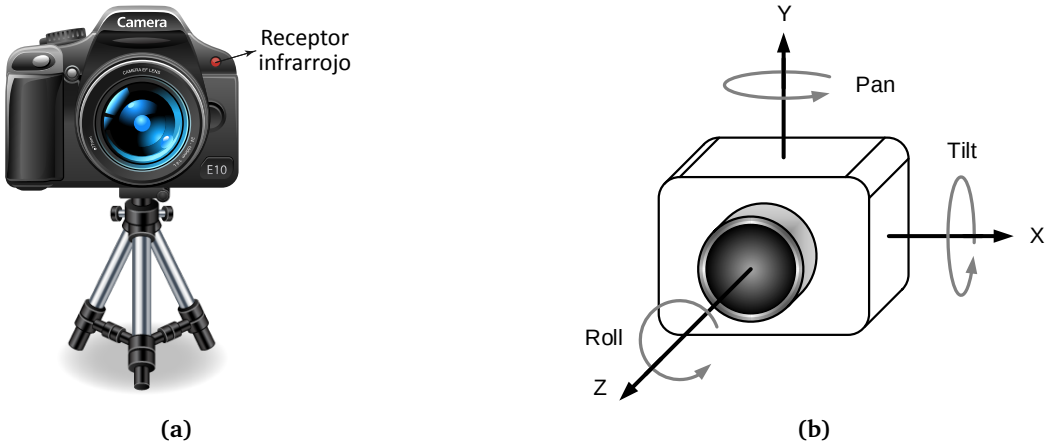


Figura 1. Montaje de la cámara sobre el trípode (a) y ángulos de rotación de una cámara (b).

Especificaciones de *hardware*

Los sensores y actuadores del sistema (véase la Figura 2) se conectarán a un microcontrolador dsPIC33FJ32MC202 de la siguiente forma:

- Pin 4: Un inclinómetro para medir el *tilt* entre –20º y 20º.
- Pines 5 y 6: Sensor de posición magnético para determinar el *pan*. El eje x se conectará al pin 5 y el z, al pin 6. Remítase al Problema 1.2 para más detalles sobre su funcionamiento.
- Pin 7: Un interruptor en configuración *pull-up* para alternar entre los dos modos de funcionamiento.
- Pin 11: Un pulsador en configuración *pull-up* para disparar el modo de funcionamiento seleccionado.
- Pin 21: Un LED infrarrojo por el que se envían trenes de pulsos para controlar la cámara remotamente.
- Pin 23: Un servomotor para ajustar el *tilt*. Para tener mayor resolución en la zona de trabajo, se añadirá una reductora de modo que 0,5 ms a nivel alto equivalgan a –20º y 2,5 ms sean 20º.
- Pin 25: Un motor de corriente continua de 3 W y 12 V nominales para modificar el *pan*.

[3]Muchas marcas tienen cámaras que cuentan con un receptor infrarrojo para accionar el obturador a distancia mediante el envío de una señal digital de unas características concretas.

Especificaciones de *software*

- El sistema contará con dos modos de funcionamiento: **automático** y **panorama**, que estarán activos al recibir un `0` o un `1`, respectivamente, por el pin 7.
- En el **modo automático**, al pulsar el disparador, la cámara se colocará perfectamente horizontal y después enviará un tren de 8 pulsos a 2 kHz al LED infrarrojo para que se tome la foto, que tarda como máximo 25 ms en realizarse (incluyendo el tiempo empleado en enviar la señal cuadrada).
- La inclinación se ajustará haciendo girar el servo tantos grados como indique el sensor de *tilt*, pero en sentido contrario. La resolución de giro del servo será de 1º. Se deberán esperar como mínimo 500 ms para que la cámara tenga tiempo de situarse en la nueva posición y estabilizarse.
- En el modo **panorama**, se repetirá la siguiente secuencia hasta que la cámara regrese a la posición inicial después de haber rotado 360º:
  1. Se corregirá cualquier desviación en la inclinación.
  2. Se enviará un tren de 8 pulsos a 4 kHz al LED infrarrojo y se esperarán 25 ms.
  3. Se hará girar la cámara 30º al 10 % de su velocidad máxima.
- Para indicar a la cámara que la toma panorámica ha terminado se enviará un tren de pulsos de 1 kHz, tras lo cual se deberá esperar al menos un segundo antes de poder realizar un nuevo disparo para dar tiempo a la cámara a construir el resultado.
- El sistema ignorará cualquier disparo que se efectúe mientras un modo esté en funcionamiento.

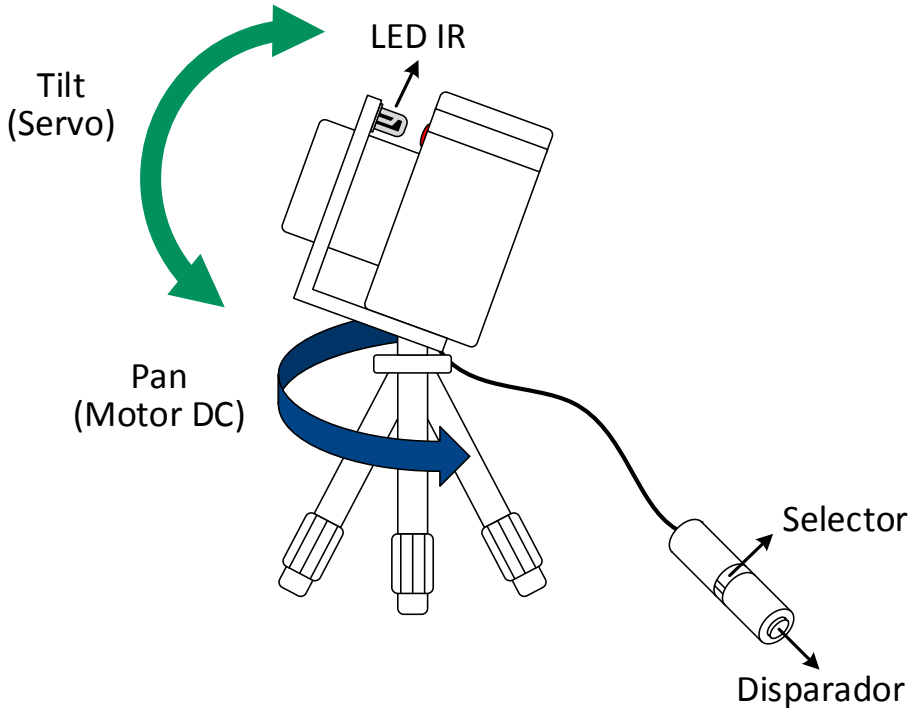


Figura 2. Diagrama del sistema con indicación de los actuadores.

Problema 1. Acondicionamiento de sensores (3,5 puntos)

✎ Resuelva este problema en el cuadernillo de color **verde**.

1.1. Inclinómetro (1,5 puntos)

Para medir el ángulo de *tilt*, se ha optado por emplear el sensor integrado `IN015` del fabricante de componentes electrónicos *NoOne, Inc*. El montaje sobre el trípode se muestra en la Figura 3.

En la práctica, se trata de un acelerómetro de un solo eje en un encapsulado de tres pines. Se alimenta entre 0 y 3,3 V por los pines 1 y 2 respectivamente y proporciona una salida proporcional a la inclinación en el pin 3. La tensión de salida es 0 V para inclinación nula, –20 mV para $\varphi = -20^{\circ}$, y 20 mV para $\varphi = 20^{\circ}$. Tenga en cuenta que el pin de salida presenta una **resistencia de salida** $R_{\text{st}} = 600\,\Omega$.

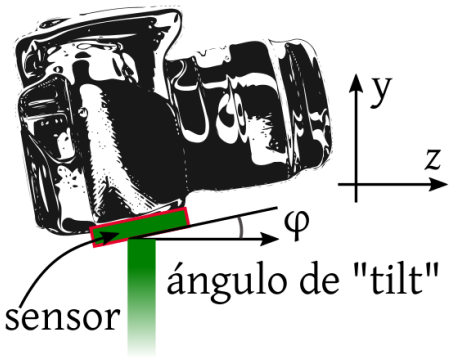


Figura 3. Montaje del inclinómetro para poder medir el ángulo de *tilt*.

Diseñe un circuito de acondicionamiento para este sensor que proporcione una salida de 1,65 V para inclinación nula y use todo el rango de 0 a 3,3 V para inclinaciones entre –20º y +20º. Puede emplear amplificadores operacionales, fuentes de tensión de valores razonables y las resistencias que necesite. Suponga que los amplificadores son ideales y pueden alimentarse con tensión simple (entre 0 y 3,3 V) o simétrica (entre –3,3 y 3,3 V); deberá indicar las tensiones de alimentación de cada amplificador.

1.2. Sensor de posición magnético (2 puntos)

El sensor de posición en el plano XZ se basa en las novedosas resistencias de efecto HBGL[4], sensores resistivos de pequeñas variaciones sensibles al campo magnético en módulo y signo. Se usarán cuatro de estos sensores (denotados S_{x1}, S_{x2}, S_{z1}, S_{z2} en la Figura 4), cuya característica es:

$$R(B) = R_0 \cdot (1 + \alpha B) \quad R_0 = 1\,\text{k}\Omega,\ \alpha = 1 \cdot 10^{-2}\,\text{T}^{-1}$$

donde B representa el campo magnético (módulo y signo) en la posición de cada uno de los sensores.

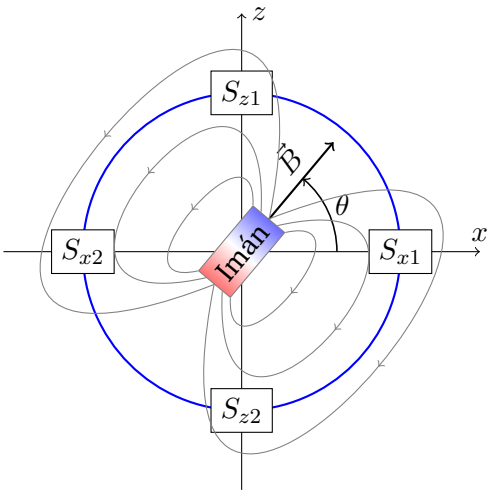


Figura 4. Sensor magnético de posición.

En el centro hay colocado un imán permanente de neodimio —cuyo campo magnético residual es muy fuerte, del orden del Tesla— que gira solidario a la plataforma donde se coloca la cámara. Dada la geometría del sistema, el campo magnético de cada uno de los sensores es:

$$B_{x1} = B_0 \cos\theta \qquad B_{x2} = -B_0 \cos\theta \qquad B_{z1} = B_0 \sin\theta \qquad B_{z2} = -B_0 \sin\theta$$

donde B_0 es el módulo del campo magnético, que en este caso resulta ser $B_0 = 0{,}1\,\text{T}$.

[4] Hall–Boal–Giannetti–Lizón.

Para acondicionar cada uno de los dos ejes se usará un circuito como el de la Figura 5 que corresponde al eje z. La salida de este circuito, V_{sz}, será proporcional al valor de $\sin\theta$, mientras que el del eje x tendrá una salida proporcional a $\cos\theta$.

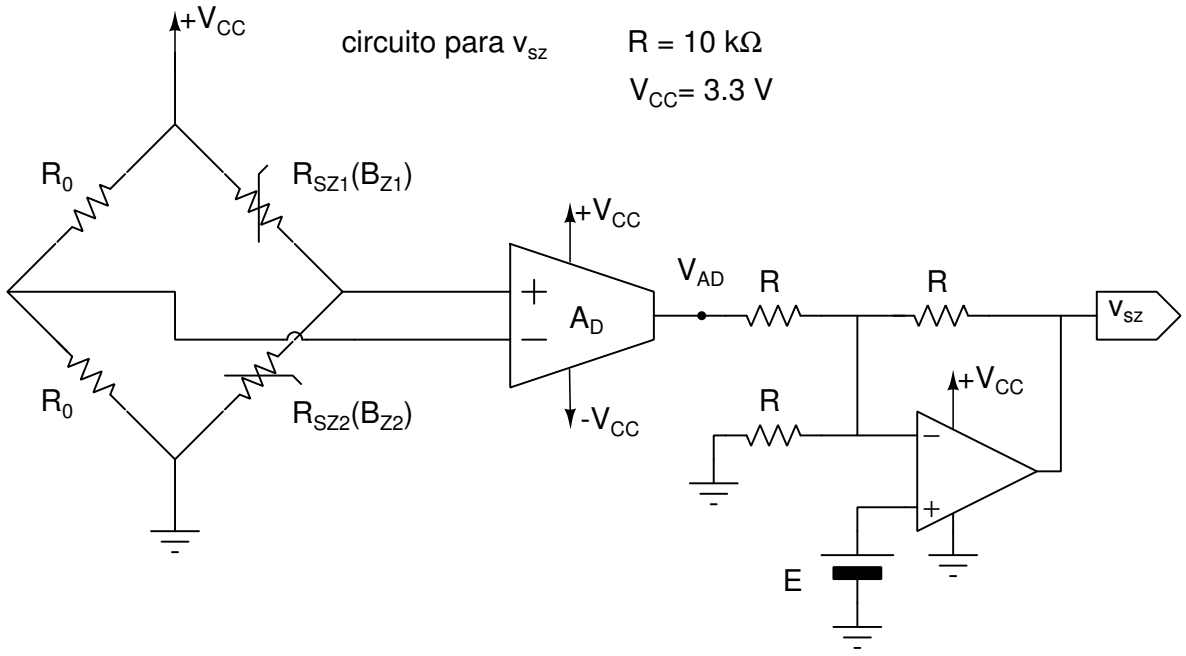


Figura 5. Circuito de acondicionamiento del sensor de posición. Se indica sólo el correspondiente al eje z; el del eje x es análogo.

a) **(1,5 puntos)** Calcule la ganancia A_D del amplificador de instrumentación y el valor de la fuente de tensión E para obtener una salida proporcional a $\sin\theta$. La tensión de salida será nula para $\sin\theta = -1$ y valdrá 3,3 V para $\sin\theta = 1$.

b) **(0,5 puntos)** Seguro que está pensando que se ha usado un puente con los sensores dispuestos de forma no tradicional. ¿Es posible colocar el sensor R_{SZ1} en otra rama del puente? En caso afirmativo, ¿cuáles son las ventajas e inconvenientes frente a la configuración actual? Considere el posible efecto de la temperatura sobre R_0, la linealidad de la tensión de salida y la flexibilidad del diseño (posibilidad de elegir el valor de la k del puente).

Problema 2. Drivers (3 puntos)

✎ Resuelva este problema en el cuadernillo de color **amarillo**.

a) **(0,5 puntos)** Dibuje de forma justificada el esquema del sistema electrónico siguiendo las especificaciones de *hardware*. Deberá incluir los circuitos de acondicionamiento de las entradas y salidas, así como su conexión con el microcontrolador. Puede emplear componentes pasivos (resistencias, condensadores, bobinas...) de los valores que necesite y transistores BJT de tipo NPN con corriente máxima de 800 mA y $\beta \in [50, 200]$. Como ya ha acondicionado previamente los sensores de inclinación y posición, puede tratarlos como cajas negras.

b) **(1,25 puntos)** Para facilitar el manejo de los sensores, programe un *driver* —no es necesario que sea genérico— usando **interrupciones** que devuelva los ángulos de *pan* y *tilt* directamente a partir de las medidas que llegan al microcontrolador. El archivo de cabecera se muestra a continuación:

sensores.h

```
#ifndef _SENSORES_H
#define _SENSORES_H

/// Configura los pines de los sensores
/// de inclinación y posición.
void inicializarSensores(void);

/// Devuelve la inclinación en el plano YZ.
/// @return Tilt en grados [-20, 20].
int getTilt(void);

/// Devuelve la posición en el plano XZ.
/// @return Pan en grados [0, 359].
int getPan(void);

#endif
```

Nota: Para usar funciones trigonométricas en C hay que incluir `math.h` (`#include <math.h>`). Todos los ángulos están expresados en radianes. Algunas funciones que podrían resultarle útiles son:

```
float sin(float x);          float asin(float x);            // Resultado: [-pi/2, pi/2]
float cos(float x);          float acos(float x);            // Resultado: [0, pi]
float tan(float x);          float atan2(float y, float x);  // Resultado: (-pi, pi]
```

Recuerde que las operaciones con `float` conllevan una mayor carga computacional, de modo que debe valorar cuál es el lugar más adecuado para realizarlas.

c) **(1,25 puntos)** Usando **interrupciones**, desarrolle un *driver* para generar los trenes de pulsos que hay que emitir por el LED infrarrojo para controlar la cámara. La interfaz será la siguiente:

disparo.h

```
#ifndef _DISPARO_H
#define _DISPARO_H

/// Configura el pin del LED infrarrojo.
/// El LED está inicialmente apagado.
void inicializarDisparo(void);

/// Genera un tren de 8 pulsos a 2 kHz
/// para realizar una única instantánea.
void disparoAutomatico(void);

/// Genera un tren de 8 pulsos a 4 kHz
/// para que la cámara tome una foto
/// de la secuencia panorámica.
void disparoPanorama(void);

/// Genera un tren de 8 pulsos a 1 kHz.
/// Inicia la combinación de imágenes.
void construirPanorama(void);

#endif
```

Problema 3. Sistema digital (3,5 puntos)

✎ Resuelva este problema en el cuadernillo de color **blanco**.

a) **(0,5 puntos)** Describa brevemente cómo piensa resolver el problema: dibuje una máquina de estados, indique cuántos timers necesita —excluyendo los que ya haya podido emplear en los *drivers*—, de qué periodo...

b) **(3 puntos)** Escriba un programa para que el sistema funcione según las especificaciones de *software*. Además de los *drivers* del Problema 2, puede utilizar el que se adjunta a continuación para controlar el motor[5].

motor.h

```
#ifndef _MOTOR_H
#define _MOTOR_H

#define PWM_L1 0  // Pin 26
#define PWM_L2 1  // Pin 24
#define PWM_L3 2  // Pin 22
#define PWM_H1 4  // Pin 25
#define PWM_H2 5  // Pin 23
#define PWM_H3 6  // Pin 21

/// Configura un pin como salida de PWM.
/// @param pin Emplear las constantes [PWM_L1, PWM_H3].
void configurarPinMotor(unsigned char pin);

/// Inicializa el generador de señales PWM a 24,743 kHz.
/// Fija el factor de servicio de los pines configurados al 0%.
void inicializarMotor(void);

/// Establece el factor de servicio de una señal de PWM.
/// @param pin Índice del pin PWM [1, 3].
/// @param dc Factor de servicio (%).
void dcMotor(unsigned char pin, unsigned int dc);

#endif
```

[5] No hay que implementar el archivo `motor.c`, sólo usar las funciones de la interfaz adecuadamente.