Departamento de Ingeniería Electrónica

Escuela Superior de Ingenieros

Evaluación continua 2019-20

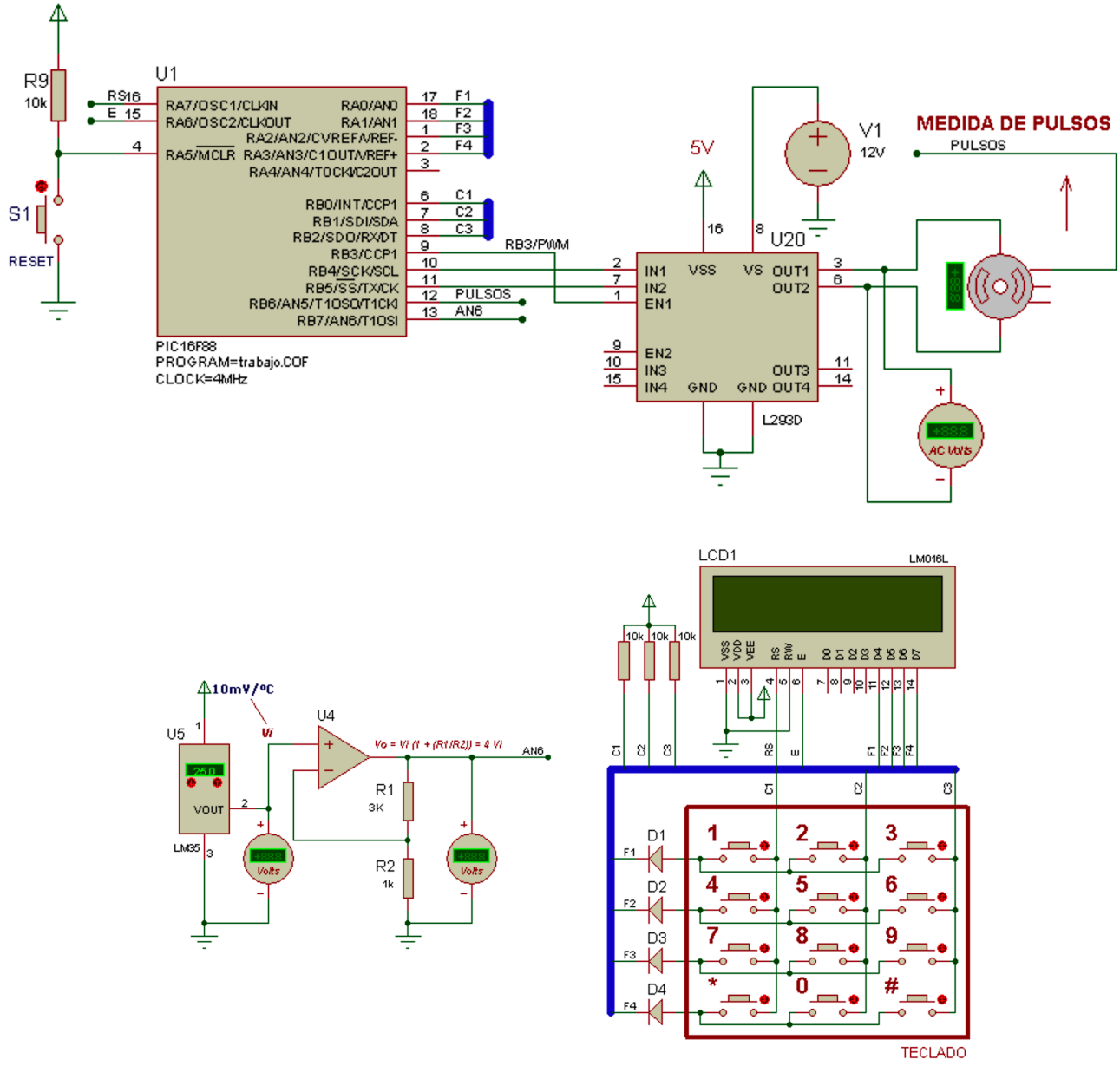
Sistemas Electrónicos Digitales

Trabajo Microcontrolador PIC16F88

Se desea diseñar un sistema de control de un motor de continua, que constará de los siguientes elementos: un teclado para introducir referencias de velocidad y realizar varias operaciones de control sobre el motor, un display LCD alfanumérico de 2x16 caracteres para mostrar información y un sensor te temperatura del motor.

Introducción al trabajo.

El esquema de conexión de los distintos elementos se muestra a continuación:

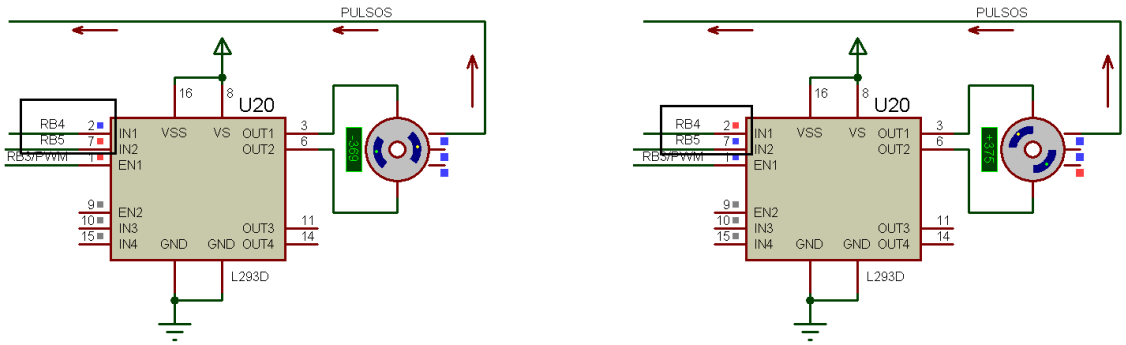


El componente L293D es un driver de potencia (concretamente dos puentes en H, del que sólo se usará uno de ellos para el motor) en el que sus salidas OUT1 y OUT2 siguen al estado de las entradas IN1 e IN2, pero con la posibilidad de dar más tensión y corriente (más potencia) para mover el motor. La señal de habilitación (EN1) es activa a nivel alto. Está conectada al pin RB3 que es la salida PWM que genera el programa. Durante el Ton (valor alto del ciclo de trabajo del PWM) el enable está activo y se ve a la salida una tensión de 12V. Durante el Toff el enable está a nivel bajo por lo que a la salida se tiene 0V (OUT1=0V, OUT2=0V). En este caso no se inyecta corriente al motor porque no hay diferencia de tensión entre sus bornas.

Sistemas Electrónicos Digitales

Trabajo Microcontrolador PIC16F88

Los pines RB4 y RB5 fijan el sentido de giro. Los valores de estos pines atacan las entradas IN1 e IN2 del L293D, respectivamente. Haciendo RB4:RB5 igual a 1:0 ó 0:1 se cambia la polaridad de las bornas del motor y, por tanto, el sentido de giro. En la siguiente figura se aprecia como cambian los valores de IN1 e IN2.



El motor dispone de un tacómetro el cual proporciona unos pulsos (señal PULSOS) según gira el eje del motor. La velocidad de giro se mide contando el número de pulsos que genera el tacómetro en una ventana temporal fija. Esa ventana temporal se ha fijado en 250 ms.

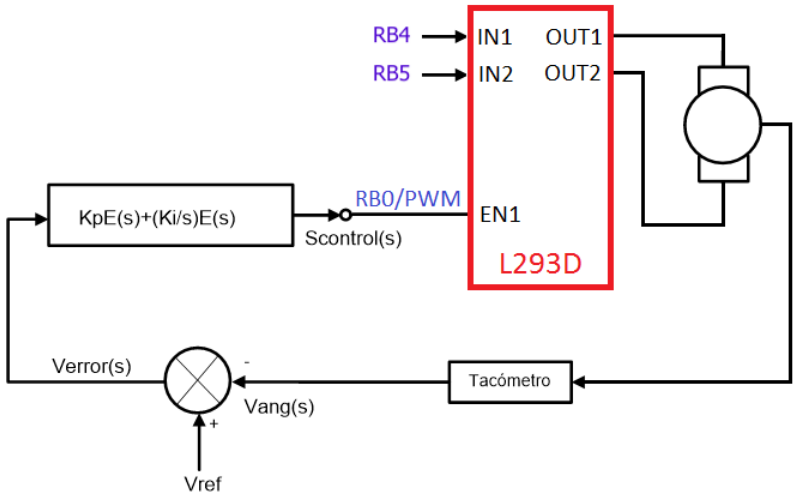
El disco del tacómetro (encoder) está dividido en 320 secciones, de manera que si se cuentan 320 pulsos (320 pulsos = 1 vuelta) en un tiempo fijado por interrupción con el Timer 0 (0.25seg), resulta que la velocidad de giro es de 320pulsos/0.25seg. Esto equivale a 1vuelta/0.25seg = 4vueltas/seg (vueltas por segundo). Aunque no sea de interés, 4(vueltas/seg)(60seg./1min.). = 240 vueltas/min (rpm). Éste último valor expresado en rpm es el que se muestra en la simulación en la cajita verde que hay a la izquierda del motor.

Por otro lado, el Timer 1 se usará como contador de pulsos del tacómetro a través del pin RB6 del micro.

Por ejemplo: si en la interrupción periódica del Timer 0 (cada 0.25s), el registro TMR1 ha contado desde 0 hasta 500, entonces se tiene que el motor va a 500/320 = 1,5625 vueltas/0,25s. O lo que es lo mismo 6,25 vueltas/s. Si este nuevo valor se multiplica por 60 (segundos/minuto), se tienen 375 rpm (revoluciones por minuto).

Dicho todo esto, aclarar que en el programa se va a trabajar con el número de pulsos que se cuentan cada 0.25seg (temporizado con el Timer 0, y acumulando pulsos con el Timer 1). El Timer 2 se usará como PWM para regular la velocidad del motor.

Los motores de continua giran a una velocidad angular que es proporcional a la corriente que circula por ellos. Para conseguir que el motor gire a la velocidad deseada, es preciso introducirlo en un lazo de control, tal y como se muestra en siguiente figura.



El tacómetro mide la velocidad de giro real del motor (Vang) y ésta se compara con la con la velocidad de giro fijada como referencia (Vref). La diferencia entre ambas magnitudes es la señal de error que se introduce en un controlador clásico de tipo proporcional-integral (PI). La velocidad de giro de referencia es el número de pulsos que el tacómetro debería alcanzar en 0.25 segundos. La velocidad de giro real será el número de pulsos que genera el tacómetro en esos 0.25 segundos y que se mide con el Timer 1 configurado como contador de pulsos:

$$Error = num\_pulsos\_ref - valor\_TMR1;$$

Los coeficientes Kp y Ki regulan la importancia de los términos proporcional (al error) e integral (la integral del error) en la señal de control. Estos coeficientes han de elegirse de manera que el sistema sea estable en todo el rango de las velocidades de trabajo y que los transitorios sean suficientemente cortos y sin sobreoscilaciones excesivas. Puesto que no se dispone de un modelo del motor, estos coeficientes se determinan experimentalmente. El valor asignado para ambos coeficientes es de 0,25 y 0,0625 respectivamente.

Puesto que se emplea un microncontrolador para regular la velocidad angular del motor, el sistema de control es en tiempo discreto, siendo el tiempo de muestreo igual a 0,25seg. La señal discreta de control se denota como S_k (la S refiere a la salida) y la señal de error como E_k (la E se refiere a la entrada, que en este caso es el error de velocidad). La señal S_k se aplicará al valor del Ton del PWM, es decir, el número que se almacenará en el registro CCPR1L. La expresión general de S_k se deduce de la siguiente manera:

Expresión del PI discreto en el instante k:

$$S_k = K_p e_k + K_i \sum_{m=0}^{k} e_m$$

Expresión del PI discreto en el instante anterior k-1:

$$S_{k-1} = K_p e_{k-1} + K_i \sum_{m=0}^{k-1} e_m$$

Restando ambas expresiones se llega a que la señal de control S_k se puede expresar como:

$$S_k = S_{k-1} + K_p(e_k - e_{k-1}) + K_i e_k$$

Es decir, el objetivo del controlador PI es ajustar el Ton del PWM (S_k) en función del valor anterior del Ton (S_{k-1}), el error actual (e_k) y el error obtenido en el ciclo anterior (e_{k-1}). El código en lenguaje C que nos da el nuevo valor del Ton en función del error entre la velocidad de referencia y la velocidad medida es el siguiente: (se proporciona ya realizado en las plantillas, archivo "interrupcion.c")

```
short int control_PI(int error)
{
    #define Kp 4                          // Definición de constantes
    #define Ki 1

    salidaPI=salidaPI+Kp*(error-errorAnt)+Ki*error;     // Expresión del PI en valores multiplicados por 16
    errorAnt=error;                                     // Se guarda el valor anterior del error
    if(salidaPI > (((int)max_duty)<<4))                 // Saturación de la salida a valor multiplicado 16
        salidaPI=(((int)max_duty)<<4);
    if(salidaPI < 0)
        salidaPI=  0;
    return(salidaPI>>4);                                // Se proporciona la salida dividida por 16
}
```

Los valores apropiados para los términos de ganancia Kp y Ki son de 0,25 y 0,0625 respectivamente. Para evitar trabajar con flotantes, se multiplican estas constantes por 16 (Kp=4 y Ki=1) y, al finalizar se divide la salida entre 16, quedando el mismo resultado, pero sin usar operaciones con flotantes. Esta división se realiza muy eficientemente con un desplazamiento ">>4". Esto hace que las ganancias Kp=4 y Ki=1, en realidad sean de 0,25 y 0,0625 respectivamente.

Se puede observar también que el control contiene la saturación (anti wind-up) usando la variable "max_duty" como valor máximo a aplicar a la salida (y 0 como valor mínimo). Este valor se encuentra multiplicado por 16 (<<4) ya que la variable salidaPI se encuentra ya multiplicada a través de las constantes Kp y Ki.

Por otro lado, en las expresiones del término integral no se ha introducido el tiempo de muestro, de manera que la constante integral Ki real sería equivalente a:

$$Ki = 0.0625 = Ki\_real * T_{PWM} = Ki\_real * 0.25$$

donde, T_{PWM} es el periodo con el que se ejecuta el PI, de lo que se desprende que Ki_real = 0.25.

Sistemas Electrónicos Digitales

Trabajo Microcontrolador PIC16F88

Desarrollo del trabajo.

En este apartado, se detallan todos los cálculos y programas que deben realizar y completar los alumnos.

Temporizaciones

Sabiendo que el microcontrolador va a trabajar con un reloj de 4 MHz, se necesita configurar los tres Timers de la siguiente forma:

Timer0: se usará para temporizar 0,25 segundos y 1 segundo. Como no hay posibilidad de que se llegue a 0,25 segundos en una sola interrupción, se necesitarán contar varias interrupciones para llegar a estos dos tiempos. El tiempo 0,25 segundos se usará para ejecutar el controlador PI (tiempo de muestreo de 0,25 segundos), y el tiempo de 1 segundo se usará para poner una bandera a 1 (tiempo_1s=1) de manera que en el bucle infinito se actualice la información mostrada por el display LCD.

Se debe inicializar en el archivo "inicializacion.c" y su interrupción se debe atender en el archivo "interrupción.c".

Timer1: se usará como contador de pulsos del tacómetro del motor. Se utiliza este timer ya que el número de pulsos en 0,25 segundos puede llegar a superar el valor de 1000 pulsos, con lo que es más cómodo usar este timer que es de 16 bits. Dentro de la interrupción del Timer 0, cuando hayan pasado 0,25 segundos, se tomará el valor del Timer 1, y se almacenará en la variable "valor_Timer1", para a continuación volver a poner el Timer 1 a 0 (para que cuente nuevamente los pulsos en otros 0,25 segundos).

Se debe inicializar en el archivo "inicializacion.c" y se debe tomar su valor en la interrupción del Timer 0, archivo "interrupción.c".

Timer2: se usará como PWM de periodo 4ms. Se deberá comprobar que el periodo del PWM es de 4ms en el osciloscopio de la simulación. Anotar el valor del periodo del Timer 2 (PR2) ya que se necesitará más adelante (es el valor máximo del duty cycle, es decir, se corresponde con un duty cycle del 100%).

Se debe inicializar en el archivo "inicializacion.c" y no se usarán interrupciones.

Adicionalmente, se deberán inicializar los registros detallados en las funciones "init_registros" e "init_librerias" con los pines correspondientes al esquema usado. Dichas funciones están en el archivo "inicialización.c".

Librerías

Se proporcionan dos librerías: TECLADO y LCD.

La librería TECLADO es una versión reducida de la usada en las prácticas. Se ha reducido ya que el programa final del trabajo es bastante extenso y podría no entrar con holgura en la memoria disponible en el microcontrolador. La funcionalidad es la misma, salvo en dos aspectos: no se incluye la librería de REBOTES, y la tecla devuelta por la librería es un número decimal, en vez de un carácter. Los valores devueltos son: 0...9 para las teclas numéricas, 10 para el asterisco y 11 para la almohadilla.

Por otro lado, la librería LCD es la habitual de las prácticas.

Todos los archivos proporcionados en las plantillas, ya disponen de los "#include" necesarios.

Máquinas de estado

En este trabajo se desarrollarán 3 máquinas de estado. En una primera etapa se hará uso de dos máquinas de estado para el manejo del teclado.

La primera máquina, archivo "procesoTeclado.c" se proporciona completa y es la habitual máquina de estados que proporciona el valor de la tecla pulsada en cada flanco de bajada del pin correspondiente. Proporciona en la variable "tecla" el valor de la tecla pulsada si la bandera "nuevaTecla" está a 1. La bandera deberá se bajada en la segunda máquina de estados.

La segunda máquina a desarrollar por los alumnos, archivo "procesoSecuencia.c", deberá recorrer las siguientes pulsaciones de teclas y realizar las acciones correspondientes al finalizar cada secuencia:

Secuencia 1: empieza por asterisco, continúa con un número (de una a tres cifras) y termina con almohadilla.

Las posibilidades son: *N# ó *NN# ó *NNN#.

Para ayudar al usuario, en la fila inferior del LCD se muestra el valor introducido.

Secuencia 2: empieza por almohadilla, se cambian los estados de dos variables pulsando 0 y/o 1, y termina con almohadilla. Para ayudar al usuario, en la fila inferior del LCD se muestran las acciones que se van a realizar. En este caso hay muchas posibilidades, ya que puede haber varias pulsaciones de 0 y/o 1.

Ejemplos:
#0# cambia el estado del motor de apagado a encendido, y viceversa.
#1# cambia el sentido de giro: sentido agujas del reloj, contrario a las agujas del reloj (CW,CCW).
#01# cambia el estado y el sentido de giro del motor.
#00# no cambia nada, ya que cambia dos veces: encendido > apagado > encendido
#001# lo mismo que #1#
…etc.

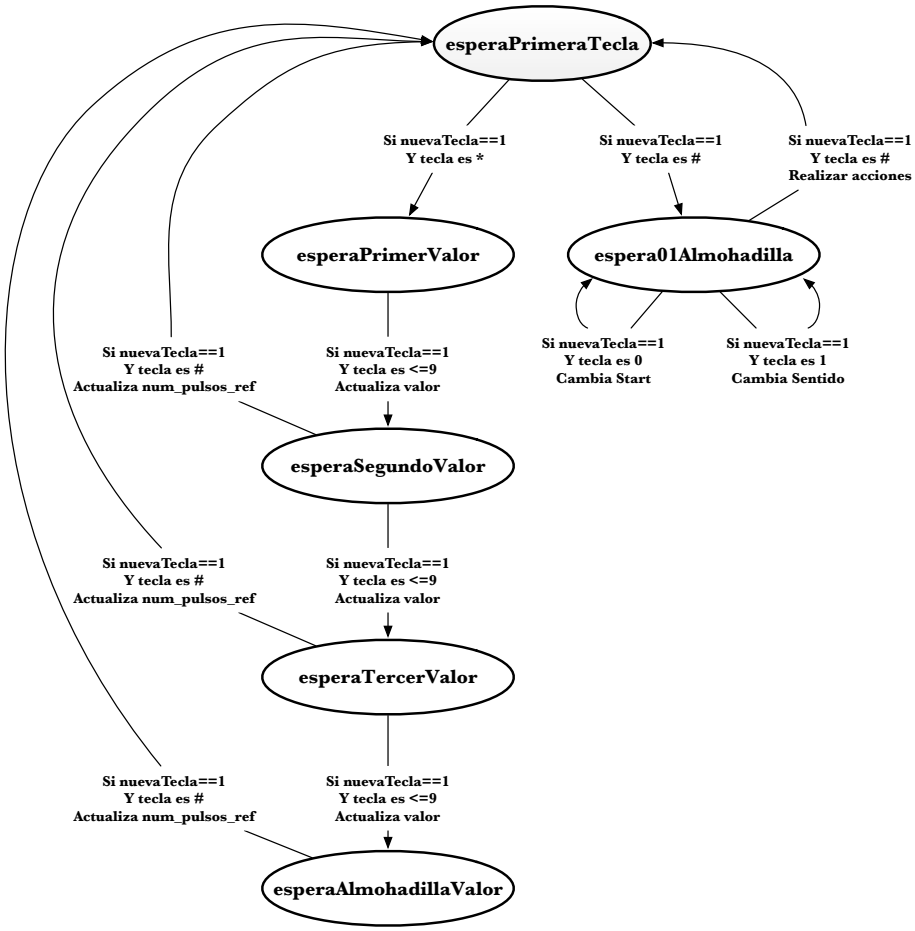
Sistemas Electrónicos Digitales

Trabajo Microcontrolador PIC16F88

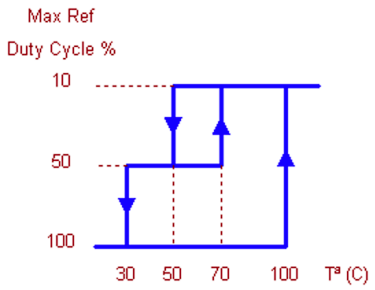
De esta segunda máquina de estados se proporciona el esqueleto de las funciones y los alumnos deben completar las acciones correspondientes a cada estado, así como las transiciones de estado.

Se deberá tener especial cuidado cuando se envíen al LCD las teclas pulsadas, ya que el LCD necesita caracteres y las teclas son números enteros (se deberá convertir de valor decimal a valor ASCII).

Para que les sirva de referencia, el esquema del diagrama de estados correspondiente a esta segunda máquina de estados es el siguiente:



La tercera máquina de estados es la correspondiente a la histéresis de temperatura medida, la cual modificará el valor del duty cycle máximo siguiendo el siguiente gráfico:



En este gráfico se puede ver que, para una temperatura superior a 100ºC, el máximo Duty Cycle expresado en tanto por ciento es del 10%. Por otro lado, para temperaturas inferiores a 30ºC, el máximo Duty Cycle es del 100%. Entre los valores 50ºC y 70ºC existen transiciones indicadas por flechas que los alumnos deben implementar, y un estado adicional correspondiente al 50%. La variable sobre la que debe escribir este proceso es "max_duty". Se debe tener en cuenta que para un valor del 100% del Duty Cycle, esta variable debe tomar el valor de PR2 configurado en el Timer 2, y para el resto se debe hacer proporcional.

El valor de la temperatura lo deberán tomar de la variable "ADC_resultado" convertida a enteros entre 0 y 1023 (10 bits de resolución). El CAD deberán configurarlo en la función "init_ADC" del archivo "inicializacion.c", y deberán realizar la conversión en el bucle infinito del archivo "trabajo.c".

Sistemas Electrónicos Digitales

Trabajo Microcontrolador PIC16F88

Se pide:

- Realizar la inicialización de los registros básicos del microcontrolador, de los Timers, y de las librerías. Completar para ello el archivo **inicializacion.c.** (2 puntos)
- Realizar la rutina de interrupción que gestiona los tiempos y la ejecución del controlador PI. Se necesitan datos calculados en el apartado anterior. Completar para ello el archivo **interrupcion.c.** (2 puntos)
- Realizar la gestión de las secuencias introducidas por el teclado mediante un proceso. Completar para ello el archivo **procesoSecuencia.c.** (2 punto)
- Realizar la gestión de la temperatura mediante la histéresis mediante un proceso. Se debe inicializar el CAD en el archivo **inicializacion.c**, descomentar la línea "(*estado[2])();" y realizar la conversión en el archivo **trabajo.c**. Se debe realizar un esquema del diagrama de estados (se entrega en un archivo PDF) detallando las transiciones y las acciones realizadas en cada estado y en cada transición, y su implementación en el archivo **procesoHisteresis.c**. Llegados a este punto, el programa estará completo en su totalidad. (4 puntos)
- Punto adicional para rematar la faena. Añadir a la máquina de estados de secuencia de teclado, la posibilidad de programar las constantes Kp y Ki. Consiste en añadir una nueva secuencia del estilo #2$N_{KP}N_{KI}$# donde N_{KP} y N_{KI} serían dos números. Dado que el programa casi con toda probabilidad no entrará en la memoria del micro, ejecuten el archivo pro.bat de la carpeta "pro" ☺, después ejecuten d.bat y vuelvan a compilar. (+1 punto, limita a 10)

A la hora de codificar hay que tener en cuenta las siguientes consideraciones:

1. Se usará el esquemático de Proteus "trabajo.dsn" y éste no podrá ser modificado.
2. Se usarán las plantillas y las librerías proporcionadas. Las plantillas contienen signos de interrogación '?' donde hace falta añadir algún código o rutina, pero el número de interrogaciones no tiene nada que ver con lo que hay que escribir (puede haber más o menos interrogaciones que cosas que escribir). Las librerías no podrán ser modificadas.
3. Se evitarán los tabuladores y se utilizará en su lugar 4 espacios en blanco para que el texto no sea dependiente de la configuración del tabulador del editor usado.

Notas sobre la entrega y consulta:

- Fecha de inicio/fin: 4 de mayo de 2020/ 31 de mayo de 2020.
- La nota del trabajo contendrá una parte que evaluará los comentarios incluidos en los archivos. **Un programa sin comentarios se penalizará restando 4 puntos sobre el total**.
- Se recuerda que se deben entregar los mismos archivos de plantilla suministrados más un pdf con la máquina de estados solicitada (histéresis). Es necesario incluir el dibujo de la máquina de estados y los comentarios que se crean oportunos para detallar cómo se resuelve lo solicitado en esta guía del trabajo. **IMPORTANTE: entregar un archivo ZIP y un PDF fuera del ZIP.**
- **La nota del trabajo será de 0 puntos** en caso de que los archivos no compilen correctamente en la **versión de Proteus 7.x** y, por tanto, no pueda simularse. **NO SE ACEPTA la versión 8 y posteriores.** IMPORTANTE: vuelve a leer este punto. Esta claro, ¿verdad? Ni otras versiones ni errores al compilar.
- **La nota del trabajo será como máximo de 3 puntos** sobre 10, si se entrega fuera de plazo.
- El trabajo se entregará por medio de la Enseñanza Virtual. **MUY IMPORTANTE**: si se hace en grupo de hasta tres personas, **TODOS LOS ALUMNOS DEBERÁN ENTREGAR** lo mismo en la Enseñanza Virtual, y además deberán dejar un comentario en la entrega con los **NOMBRES DE TODOS LOS ALUMNOS**. Se permite la entrega individual, pero también dejando el comentario en la entrega.
- Para evitar confusiones, el trabajo sólo lo tutorará el profesor Juan Antonio Sánchez (juansanchez@us.es) y se aceptan en principio sólo consultas por correo electrónico, aunque el profesor está abierto a otras posibilidades online.
- **Notas importantes a cerca de las consultas**:
    1. No se atenderán consultas del tipo "no funciona", "me da un error" ó "no me da error al compilar, pero no funciona". Los alumnos deberán depurar su programa (que para eso está el simulador) y entender y buscar solución a los problemas planteados. Si se aceptarán dudas sobre la interpretación de lo expuesto en esta guía.
    2. No se atenderán consultas del tipo "no me compila" sin que los alumnos hayan entendido el mensaje de error proporcionado por el compilador. El 99% de las consultas de este tipo se resuelve **LEYENDO** la ventana de salida del compilador. Los emails con capturas de pantalla en la que se pueda leer el error no serán contestados.
    3. En ningún caso, se enviará código al profesor para que éste diga qué está bien y qué falla. Los alumnos deben ser capaces de revisar su propio código y el suministrado en las plantillas. Sólo se aceptarán dudas puntuales en el código y además estas dudas deberán ser clara e inequívocamente especificadas y razonadas.