

UNIVERSIDAD NACIONAL DE CÓRDOBA

FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES

CATEDRA DE ELECTRÓNICA DIGITAL II

TRABAJO PRÁCTICO FINAL

**“REGULADOR DE TEMPERATURA A PEQUEÑA**

**ESCALA”**

Alumnos:

Felizia, Valentín

Ledesma, Ignacio

Profesor:  
Ing. Vrech, Rubén

Comisión: Miercoles – 9hs

Junio / 2024

**Consigna**

Diseñar y construir un dispositivo electrónico basado en el PIC16F887, donde mínimamente se utilicen las funciones de **interrupción**, **Timer0**, **ADC** y **comunicación en serie** propias del microcontrolador, y además deberá contar con un **teclado** para ingresar datos o parámetros y al menos **2 displays** para mostrar datos o parámetros.

**Desarrollo**

1. **Selección de la idea:**

El primer paso fue determinar qué proyecto creativo, útil y simple podíamos llevar a cabo cumpliendo los requerimientos establecidos en la consigna. En primera instancia pensamos realizar algo utilizando un sensor ultrasónico, pero no se nos ocurrió como implementar un teclado en ese caso y que haga algo útil, después pensamos en un sistema de control de humedad o un medidor de nivel de líquidos, pero no eran ideas muy creativas ya que son proyectos bastante frecuentes.

Lo que se nos ocurrió al final fue hacer un sistema a pequeña escala que refrigera la temperatura de un líquido hasta una temperatura deseada (la cual se puede configurar en tiempo real), simulando sistemas de refrigeración líquida, pero para agregarle un poco más de complejidad el profesor nos recomendó implementar algo que también caliente el agua de ser necesario. Así llegamos a la idea final: un sistema de regulación de temperatura.

1. **Programación y Simulación:**

Una vez definida la temática del proyecto, el siguiente paso fue empezar el desarrollo del código con el que se iba a programar nuestro PIC16F887. Nuestro acercamiento a esta etapa fue plantear distintos hitos, o milestones en inglés, a alcanzar a lo largo del proyecto, dividiendo la programación en etapas y a lo largo de cada etapa también íbamos avanzando con la simulación. Primero analizamos como debíamos configurar al PIC16F887 (esto incluye calcular como debíamos precargar al Timer0 para hacer una multiplexación adecuada de los displays, por ejemplo), luego implementamos el multiplexado para el teclado y el display, después desarrollamos el código para la conversión a través del ADC de la señal analógica de temperatura a digital (y que dicha temperatura se muestra en el display). Lo siguiente fue hacer el código para configurar la temperatura deseada a través del teclado, y esta fue la parte mas compleja ya que hubo que tener en cuenta aspectos como el rebote de tecla o que no se ingrese el mismo número múltiples veces si mantengo apretada la tecla. Finalmente, desarrollamos el código en Assembly para la transmisión en serie y también hicimos un programa en Python capaz de interpretar los datos transmitidos en serie desde el PIC.

Por último, implementamos el código completo dentro de la simulación que fuimos construyendo y comprobamos que todo funcionara adecuadamente.

1. **Topología (Cálculos) del Circuito:**

Como ahora ya sabemos y corroboramos el funcionamiento óptimo del circuito a través de la simulación, la próxima etapa es comenzar a idearlo desde el punto de vista puramente electrónico.

Primero definimos que la tensión con la que vamos a alimentar el circuito van a ser 12V, que es lo necesario para alimentar al ventilador y a la lampara incandescente que utilizamos y que luego se reducen a 5V para el PIC, el sensor de temperatura y la bomba de agua. Además, es importante identificar todos los componentes que vamos a necesitar para buscar sus planillas de datos y verificar si es necesario plantear ecuaciones que los tengan en cuenta.

Finalmente, se realizan los cálculos necesarios para la construcción del circuito, como los valores de las distintas resistencias utilizadas. Con todo esto en mente, se realiza la compra de todos los componentes a utilizar.

1. **Construcción y Validación del Circuito:**

Habiendo verificado la electrónica del circuito, se lo construye en una protoboard y se comprueba si su funcionamiento es el deseado. Luego de verificar que todo funciona bien, se procede con la construcción del contenedor y de la estructura del circuito, lo que incluyó hacerle una sonda al sensor de temperatura para poder sumergirlo bajo el agua.

**Cálculos**

* 1. **Precarga del Timer0**:

En nuestro programa utilizamos al Timer0 para hacer el multiplexado de los displays y del teclado. Si bien son solamente 2 displays, se piensa como si fueran 3 debido a que se hace el multiplexado de 3 columnas del teclado también.

Teniendo en cuenta que el ojo humano no detecta variaciones superiores a 50Hz, eso equivale a encender un display cada 20ms (1/50Hz), y como el multiplexado se hace como si fueran 3 displays, significa que cada display debe permanecer encendido 6,67ms (20ms/3) mínimamente.

Con esto en mente, podemos calcular con qué valor se precarga al Timer0, suponiendo una frecuencia del Clock de 4MHz (que es el valor por default) y habiendo configurado previamente un Preescaler de 64 (esta fue una elección arbitraria, no por nada en específico).

* 1. **Resolución del ADC**

Para convertir correctamente la señal analógica recibida a través del sensor a números visibles en el display, es necesario adaptarse a la resolución que posee el ADC del PIC, para ello se realiza el siguiente cálculo, teniendo en cuenta que el resultado del ADC ocupa 10 bits:

Observando este resultado y analizando la datasheet del LM35, se distingue que la señal analógica devuelta por el sensor sigue la siguiente ecuación:

*T es la temperatura en grados Celsius*

Como nosotros suponemos que la temperatura va a permanecer entre 0 y 99 ºC, el máximo numero que va a mostrar el ADC es 11000110, lo que nos indica que vamos a tener que justificarlo a la derecha, priorizando ADRESL (si justificáramos hacia la izquierda, eso haría que 2 bits relevantes quedaran fuera del ADRESH). Además, como la resolución del ADC es de 5mV y la tensión entregada por el sensor aumenta cada 10mV, eso quiere decir que el bit menos significativo del ADC no importa realmente, y esto en el código se tiene en cuenta (se hace una rotación a la derecha del resultado y después se opera con eso)

* 1. **Resistencias para los segmentos del Display:**

Nosotros optamos por utilizar displays de cátodo común, pero se pueden usar de ánodo común también (de ser así, lo correcto sería cambiar el tipo transistor que se utiliza para el multiplexado de los displays y el número en binario que se utiliza para la multiplexación dentro del programa).

Si analizamos la datasheet del display que utilizamos, se observa que cada segmento consume aproximadamente 2V. Además, nosotros optamos por limitar la corriente que reciben a 15mA ya que no es necesario exigirle más corriente al PIC para el fin de nuestro proyecto (cabe destacar que no se utilizan transistores en la conexión entre el PIC y los segmentos justamente porque no es necesario alimentar a los segmentos con más de 25mA, que es lo que puede llegar a entregar un pin I/O del PIC, solamente se requieren más de 10mA).

De esta forma, en nuestro caso, tanto los segmentos como los displays se encenderán por lógica positiva. Con esto en mente, podemos plantear la siguiente ecuación siguiendo la Ley de Ohm:

Analizando este resultado, decidimos utilizar resistencias de para los segmentos del display.

* 1. **Resistencias para el Transistor que enciende al Display**

Como se mencionó anteriormente, utilizamos displays de cátodo común, por lo que lo correcto es conectar el cátodo común de cada display a un transistor NPN. Este transistor tendrá su base conectada a los pines RB6 o RB7 del PIC, el colector a los displays y el emisor a tierra. Utilizar un transistor PNP y lógica negativa para los displays no es posible al utilizar display de cátodo común porque habría que conectar el emisor del transistor al display (ya que la corriente va del display a la tierra), y si la caída de tensión en la carga es mayor que la disponible en la malla base-emisor, que en este caso es lo que sucedería, el transistor nunca podría llegar a saturarse y, por lo tanto, no podría encenderse el display.

La idea de realizar esta conexión es que, cuando el PIC envíe un 1 por los pines RB6 o RB7, el transistor se sature y conecte la corriente proveniente del display con la tierra, permitiendo que el display se encienda. Así, podemos plantear lo siguiente, analizando la datasheet de nuestros transistores:

*Se tiene en cuenta el caso en el que se encienden los 7 segmentos:*

*Se supone una de 25 para asegurar que el transistor se sature correctamente:*

*Teniendo en cuenta que la tensión base-emisor en saturación es 0,7V:*

Analizando este resultado, decidimos utilizar resistencias de para la base de los transistores del display.

* 1. **Resistencia para el Transistor que enciende la Bomba de Agua**

Teniendo en cuenta que los módulos relé que compramos para el ventilador y la lámpara incandescente se activan por lógica negativa, decidimos que con la bomba de agua, que usa 5V, debería pasar lo mismo. A su vez, aunque optaramos por utilizar lógica positiva, no hubiésemos podido conectar directamente la bomba al PIC porque requiere 200mA para funcionar óptimamente a 5V, y un puerto del PIC solamente es capaz de entregar 25mA, por lo que el uso de un transistor era inevitable.

Al usar lógica negativa, es necesario implementar un transistor PNP cuya base está conectada al puerto RA1 del PIC, el colector a la bomba y el emisor a un regulador de tensión que devuelve 5V. Para lograr que el transistor se sature adecuadamente y permita el paso de la corriente del emisor al colector, es necesario calcular qué resistencia poner en la base de este, y para ello se plantea la siguiente ecuación:

*Se supone una β mínima de 250:*

*Multiplicamos el resultado por un factor de 5 para asegurar una saturación completa:*

*Teniendo en cuenta que la tensión base-emisor en saturación es 0,7V:*

Analizando este resultado, decidimos utilizar resistencias de para la base del transistor de la bomba de agua.

**Diagramas Circuitales**

Diagrama, Esquemático

Descripción generada automáticamente

Figura 1: Diagrama Circuital (Fuente: Proteus)

**Materiales**

* **12V:** cualquier fuente de 12V con 3A mínimo (si es regulable mejor, el sensor de temperatura se puede quemar sino)
* **U1:** Regulador de Tensión LM7805
* **U2:** Sensor de Temperatura LM35
* **U3:** PIC16F887
* **RESET:** botón push
* **RL1, RL2:** módulos con relé SRD-05VDC-SL-C
* **PER1:** ventilador genérico de 12V
* **PER2:** lampara incandescente de 12V
* **PER3:** mini bomba de agua de 5V
* **PER4 y PER5:** displays 7 segmentos de cátodo común
* **PER6:** modulo adaptador RS232TTL DB9 con Max232 incorporado (y para conectarlo a la PC, un adaptador DB9 a USB)
* **PER7:** teclado matricial
* **R1, R5, R6, R7 y R8:** resistencias de 10 [kΩ].
* **R3, R4, R9, R10, R11, R12, R13, R14 y R15:** resistencias de 220 [Ω]
* **R2:** resistencia de 1 [kΩ]
* **T1**: transistor BC327-40
* **T2, T3**: transistores BC337-25

**Circuito Físico**

Diagrama

Descripción generada automáticamente

**Simulación y Funcionamiento**

El circuito constantemente hace multiplexado del teclado y los displays. Se inicia estableciendo una temperatura deseada de 26 ºC y muestra “HI” en el display hasta que se realiza la primera interrupción por Timer1, que es cuando se realiza el primer muestreo de la señal analógica recibida a través del sensor de temperatura. Una vez realizado el muestreo, se configuran los dígitos del display para mostrar la temperatura sensada y se analiza si hay que prender o no los periféricos correspondientes (si la temperatura es mayor que la deseada, se encienden los ventiladores y la bomba, de lo contrario se enciende la lampara incandescente, y en el caso de que la temperatura sea la deseada, se apagan dichos periféricos).

Por otro lado, cada vez que se presione una tecla se refleja dicha acción en los displays por un breve momento (hasta que se realice el siguiente muestreo), mostrando el valor actual ingresado por teclado o mostrando un mensaje (Er de erased al apretar \*, que hace que se borre lo ingresado por teclado, o el valor de temperatura deseada ingresado por teclado al presionar #). Cada vez que se presione #, estableciendo una nueva temperatura deseada, se envía dicha temperatura a través del puerto de transmisión en serie TX en forma de un numero binario. Si se conecta el PER6 a una computadora con el programa en Python desarrollado y ejecutado, recibe el valor establecido y lo muestra en pantalla. La simulación cumple los estándares que habíamos planeado y funciona adecuadamente. En la practica el funcionamiento también resulta ser el esperado.

**Conclusiones**

A través de este proyecto, hemos aprendido a aprovechar las capacidades del PIC16F887, como su ADC y el manejo de interrupciones para interactuar con un teclado matricial, por ejemplo. La programación en Assembler nos permitió escribir código optimizado y entender a fondo la arquitectura interna del microcontrolador. Además de fortalecer nuestra habilidad técnica, esta experiencia nos proporcionó una comprensión más profunda de los principios de la electrónica digital y la integración de sistemas hardware-software. En conjunto, esta experiencia nos ha preparado para enfrentar desafíos más complejos en futuros proyectos de diseño electrónico y sistemas embebidos, consolidando nuestro conocimiento teórico con habilidades prácticas que son fundamentales en el campo de la ingeniería electrónica.