



Universidad Nacional Autónoma de México

Facultad de Ingeniería
División de Ingeniería Eléctrica
Ingeniería en Computación



Proyecto 1 “Sistema Mínimo”

Integrantes:

- Castelan Ramos Carlos
- Castillo Montes Pamela
- Hernández Jaimes Rogelio Yael

Materia: Microcomputadoras

Grupo: 01

Semestre: 2023-2

Fecha de entrega: 29 de marzo 2023



Reporte Proyecto 1

“Sistema Mínimo”

Circuito Alambrado

El sistema mínimo está formado por:

1. Microcontrolador PIC16F877A
2. Cristal de cuarzo 20 MHZ
3. 2 capacitores cerámicos de 22pF
4. 1 push uttom de 2 pines
5. 1 resistencia 10 KΩ
6. 2 capacitores cerámicos 0.1 μF
7. Alambre/Jumpers
8. Protoboard
9. Fuente de poder 5 Volts

Al sistema mínimo se le añadió una interfaz serial haciendo uso de:

1. Transmisor Serial UART/TTL con salida USB.

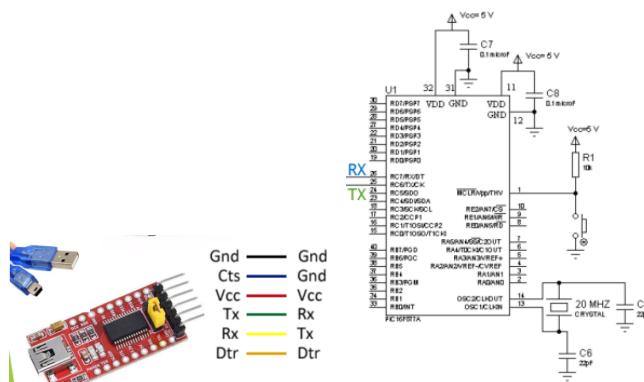


Imagen 2: Sistema mínimo

Para el desarrollo del alambrado físico del sistema mínimo, únicamente se hizo del diagrama proporcionado por el profesor.

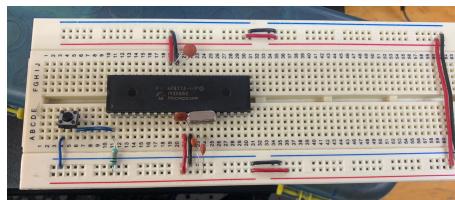


Imagen 2: Alambrado de PIC

Enseguida, para comprobar el funcionamiento del proyecto, el profesor planteó generar un programa en lenguaje ensamblador, el cual controla el encendido y apagado de leds colocados en un cartel.

Para ello primero se planteó el acomodo físico de los leds en un pequeño dibujo, así también se decidió asignar un bit de algún puerto de salida para cada led, además se decidió colocar la palabra UNAM en el cartel.

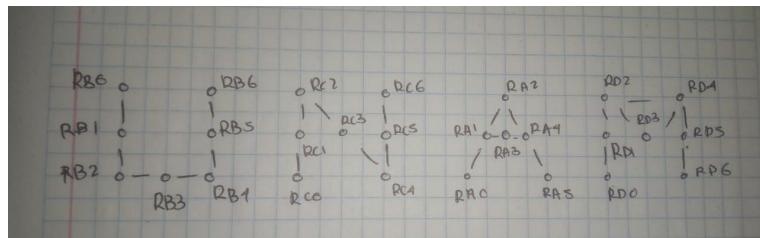


Imagen 3: Idea básica

Una vez planteada la idea básica, se generó el resultado físico en un pedazo de cartón, donde se colocaron cada uno de los leds de cada letra, para cada led en el cátodo se asignó un alambre o pin que se dirige a su respectiva salida en el PIC, para los ánodos de los leds se puentearon y se conectaron a tierra en el circuito.

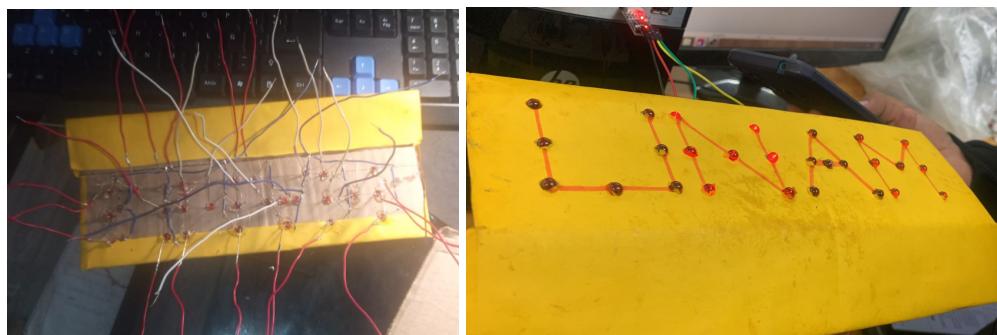


Imagen 4: Letrero físico alambrado

Finalmente se llevó a cabo la conexión del circuito mínimo, el comunicador serial, la fuente de poder de 5V y el letrero de leds.

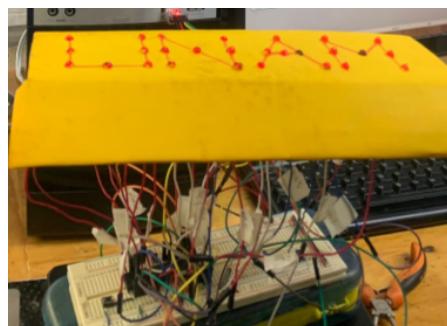


Imagen 5: Letrero físico alambrado

Programa ASM

El programa para el controlador pic16f877 implementa una secuencia de encendido y apagado para los leds. Como se mencionó previamente, cada led tiene un bit asignado a algún puerto paralelo configurado como salida, por lo que para generar las secuencias tendremos que activar o apagar cada uno de estos.

En esta ocasión, implementaremos dos secuencias sencillas: apagado-encendido y rotación a la izquierda. Antes de esto, necesitamos definir algunas instrucciones adicionales. Por ejemplo, la línea 1 y 2 de forma general permiten “cargar” las configuraciones y directivas definidas para el controlador PIC17F877 (como el nombre de los registros específicos). Así



mismo, definimos algunas variables y constantes. El registro AUX será utilizado para contabilizar la iteración en ejecución. Con esta accederemos de forma automática a las diferentes secuencias según el valor que tenga. Los registros REGA, REGB y REGC son registros auxiliares para la subrutina de retardo. En estos realizamos las operaciones necesarias para retrasar la ejecución. Los valores constantes Aval, Bval y Cval definen los valores operantes en la subrutina de retardo. A mayor valor de estos mayor es el tiempo que genera de retardo. Los valores definidos generan un retraso aproximado de 1 segundo. Esto responde directamente a la fórmula:

$$\text{Ciclos} = C[B(3A + 1) + 3B + 1] + 3C + 3$$

$$\text{Tiempo} = \text{Ciclos} * (0.2\mu\text{s})$$

Donde si sustituimos los valores de A, B y C vemos un tiempo aproximado de 1 segundo. Las líneas 12, 13 y 14 definen el vector reset, el direccionamiento a la etiqueta INICIO y el establecer el origen del programa en 5h (a partir de donde se definen las instrucciones consecuentes) respectivamente.

```
1      PROCESSOR 16F877
2      INCLUDE <P16F877.INC>
3
4      AUX equ h'24'
5      REGA equ h'25'      ;Registros auxiliares para rutina de retardo
6      REGB equ h'26'
7      REGC equ h'27'
8      Aval equ .255       ; Valor de A en subrutina de retardo
9      Bval equ .255       ; Valor de B en subrutina de retardo
10     Cval equ .76        ; Valor de C en subrutina de retardo
11
12     ORG 0 ;Vector de reset
13     GOTC INICIO
14     ORG 5
```

Para el inicio del programa (etiqueta INICIO) procedemos a configurar los puertos paralelos. Para ello primero limpiamos el contenido de los registros PORT*, para evitar cualquier situación. Dado a que estamos en el banco 0, hacemos un cambio al banco 1 al establecer RP0 en 1 y RP1 en 0. Antes de cualquier configuración definimos los puertos A y E como puertos digitales, para lo que el registro ADCON1 movemos el valor 06h, como lo especifica la datasheet del microcontrolador.

Estando en el banco 1 configuramos los registros TRIS* como salidas, por lo que todos estos movemos el valor 0h, o que implica que todos los bits de TRIS* se encuentren en 0. Tras esto, procedemos a regresar al banco cero, esto asignando 0 a RP0. Ya en el banco cero nuevamente comienza la secuencia. El LOOP inicial inicializa al contenido del registro AUX en 0. Recordemos que AUX es un contador para hacer los cambios de secuencias. El LOOP_1 corresponde al Loop interno de la secuencia para encendido-apagado. Lo que se hace en este Loop es mover el valor de AUX al registro W para poder operar con el valor. Lo que se hace es restar 6 a W, si es igual la bandera Z se activará. La línea 38 comprueba si la bandera Z está activa, pues significa que AUX tiene valor de 6 y por tanto la secuencia encendido-apagado se ha realizado 6 veces. Si es así, entonces no pasa a la secuencia sino que pasa al LOOP_2.



```

15    INICIO: CLRPF PORTA ; Limpia PORTA
16    CLRPF PORTB ; Limpia PORTB
17    CLRPF PORTC ; Limpia PORTC
18    CLRPF PORTD ; Limpia PORTD
19    BSF STATUS,RPO ; Cambia a banco 1
20    BCF STATUS,RP1
21    MOVLW 0EH ; Define puertos A y E como digitales
22    MOVWF ADCON1
23    MOVLW H'0'      ; CONFIGURA PUERTO A COMO SALIDA
24    MOVWF TRISA     ; (TRISA) <-- 0h
25    MOVLW H'0'      ; CONFIGURA PUERTO B COMO SALIDA
26    MOVWF TRISB     ; (TRISB) <-- 0h
27    MOVLW H'0'      ; CONFIGURA PUERTO C COMO SALIDA
28    MOVWF TRISC     ; (TRISC) <-- 0h
29    MOVLW H'0'      ; CONFIGURA PUERTO D COMO SALIDA
30    MOVWF TRISD     ; (TRISD) <-- 0h
31    BCF STATUS,RPO ; Cambia al banco 0
32
33    LOOP:           MOVLW H'0'      ; W<-- 0. Inicializa AUX (contenido) en 0
34          MOVWF AUX     ; (AUX)<-- 0
35    LOOP_1:           ; Ejecuta secuencia encendido-apagado seis veces
36        MOVF AUX,W   ; W<--(AUX)
37        SUBLW 0x06     ; W<--W-0x06
38        BTFSS STATUS,Z ; (CONTA)=0X06?
39        GOTC ONOFF    ; NO
40        GOTC LOOP_2    ; SI

```

El LOOP_2 es casi igual que LOOP_1, sin embargo, ahora se evalúa si el valor de AUX es 9, que significaría que se ejecutó la secuencia de “rotación a la izquierda” 3 veces. Además, este Loop pasa a la secuencia ROTAIZQ.

```

41    LOOP_2:           ; Ejecuta secuencia de rotación a la izquierda 3 veces
42        MOVF AUX,W   ; W<--(INDF)
43        SUBLW 0x09     ; W<--W-0x09
44        BTFSS STATUS,Z ; (CONTA)=0X09?
45        GOTC ROTAIZQ  ; NO
46        GOTC LOOP      ; SI

```

La secuencia encendido-apagado se define a partir de la etiqueta ONOFF. Aquí únicamente activamos todos los bits de cada puerto paralelo (del puerto A al puerto D). Ahora bien, cada letra está asignado a un único puerto, pero no usan todos los bits disponibles del puerto. Por ejemplo, La letra N no emplea el bit RC7. Debido a esto es que el valor que se asigna a cada puerto (a través de W) es el valor hexadecimal correspondiente a activar únicamente el número adecuado de leds. Por ejemplo, el puerto A tiene asignado 3F, lo que en binario es 00111111, como puede verse, activamos los seis bits del puerto para la letra A. De forma similar el resto de los puertos se configura con el valor 7F, para activar así los siete leds necesarios. Tras activar todos los puertos generamos un retardo de un segundo llamando a la rutina de retardo. Tras este tiempo, apagamos todos los puertos con CLRF. Tras cada secuencia aumentamos en uno a AUX, lo que implica que se iteró una vez la secuencia. Con esto el Loop puede verificar si se cambia de secuencia o no.

```

48    ONOFF:           ; Enciende y apaga todas las letras con retardo de 1 segundo
49        MOVLW H'3F'   ; Configura activar todas las salidas de A
50        MOVWF PORTA    ; (PORTA)<-- 3F -> 0011 1111
51        MOVLW H'7F'   ; Configura activar todas las salidas de B
52        MOVWF PORTB    ; (PORTB) <-- 7F -> 0111 1111
53        MOVLW H'7F'   ; Configura activar todas las salidas de C
54        MOVWF PORTC    ; (PORTC) <-- 7F -> 0111 1111
55        MOVLW H'7F'   ; Configura activar todas las salidas de D
56        MOVWF PORTD    ; (PORTD) <-- 7F -> 0111 1111
57        CALL RETARDO   ; Llamada a la subrutina de retardo (1 segundo aprox.)
58        CLRF PORTA    ; Limpia PORTA
59        CLRF PORTB    ; Limpia PORTB
60        CLRF PORTC    ; Limpia PORTC
61        CLRF PORTD    ; Limpia PORTD
62        CALL RETARDO   ; Llamada a la subrutina de retardo (1 segundo aprox.)
63        INCF AUX,F    ; AUX<-- AUX+1
64        GOTC LOOP_1;

```

La secuencia de “rotación a la izquierda” enciende cada una de las letras de izquierda a derecha. Se define a partir de la etiqueta ROTAIZQ. Siguiendo la misma configuración para encender las letras para la secuencia anterior. Sin embargo, en este caso encendemos primero la letra U (puerto B, valor 7F), generamos un retardo, lo apagamos y encendemos



la letra N (puerto C, valor 7F). De forma análoga vamos encendiendo la siguiente letra y apagando la anterior. Esto se repite consecutivamente hasta generar todo el recorrido de izquierda a derecha. Al final incrementamos en uno a AUX.

```
65    ROTAZQ:      ; Enciende y apaga de letra en letra de izquierda a derecha (1. U   2. N   3. A   4. M)
66    MOVlw H'7F'    ; Configura activar todas las salidas de B
67    MOVwf PORTB   ; (PORTB) <-- 7F -> 0111 1111  ENCIENDE LETRA U
68    CALL RETARDO
69    CLRw PORTB   ; Limpia PORTB  APAGA LETRA U
70    MOVlw H'7F'    ; Configura activar todas las salidas de C
71    MOVwf PORTC   ; (PORTC) <-- 7F -> 0111 1111  ENCIENDE LETRA N
72    CALL RETARDO
73    CLRw PORTC   ; Limpia PORTC  APAGA LETRA N
74    MOVlw H'3F'    ; Configura activar todas las salidas de A
75    MOVwf PORTA   ; (PORTA) <-- 3F -> 0011 1111  ENCIENDE LETRA A
76    CALL RETARDO
77    CLRw PORTA   ; Limpia PORTA  APAGA LETRA A
78    MOVlw H'7F'    ; Configura activar todas las salidas de D
79    MOVwf PORTD   ; (PORTD) <-- 7F -> 0111 1111  ENCIENDE LETRA D
80    CALL RETARDO
81    CLRw PORTD   ; Limpia PORTD  APAGA LETRA M
82    INCF AUX, F    ;AUX<-- AUX+1
83    GOTO LOOP_2
```

La subrutina de retardo es la vista en clase haciendo uso de tres variables. En esencia, cada Loop hace un decremento hasta 0 de el valor A, B o C. Sin embargo, por cada decremento de B se hace un ciclo completo de decremento de B y para cada decremento de B se hace un ciclo completo de decremento a 0 de A. De esta forma el controlador gasta ciclos de reloj. De forma estimada se emplea la fórmula indicada con anterioridad. Con los valores definidos por Aval, Bval y Cval se estima un retardo de aproximadamente un segundo.

```
85    RETARDO:MOVlw Cval ;Subrutina de retardo de aproximadamente 1 segundo
86        MOVwf REGC
87    LOOPC:  MOVlw Bval
88        MOVwf REGB
89    LOOPB:  MOVlw Aval
90        MOVwf REGA
91    LOOPA:  DECFSZ REGA
92        GOTO LOOPA
93        DECFSZ REGB
94        GOTO LOOPB
95        DECFSZ REGC
96        GOTO LOOPC
97    RETURN
98
99
```

Imagen 6: Programa

Conclusiones:

Castelan Ramos Carlos

Dentro del desarrollo de este proyecto, logramos implementar correctamente el set de instrucciones del PIC16F788A, generando un entendimiento en el funcionamiento de los elementos de puertos de salida o entrada a partir de sus puertos de configuración TRISX, realizando operaciones sencillas como desplazamientos entre bancos de memoria. Así también logramos realizar otros efectos en el PIC como realizar retardos consumiendo ciclos, dado que la frecuencia que maneja es demasiado alto para el ojo humano, es entonces que pudimos observar el funcionamiento de nuestro programa. Para la parte del alambrado del circuito mínimo, fue sencillo ya que únicamente se hizo uso del diagrama proporcionado por el profesor, para la interfaz serial se ocupó un transmisor serial UART/TTL el cual nos ayudó bastante ya que con este mismo realizamos la comunicación de carga de programa desde una a computadora al PIC además de que otorgaba los 5V necesarios para el funcionamiento del PIC. Por otra parte, el circuito de cartel de leds para probar el correcto funcionamiento de nuestro microcontrolador funcionó correctamente también, en donde para cada led se le asignó un pin de salida en los bits de los puertos y la tierra se puentearon entre cada uno, finalmente considero que se cumplió con el con éxito el armado, funcionamiento y puesta a prueba del proyecto.



Castillo Montes Pamela

Durante la realización de este proyecto, se tuvo un primer acercamiento a los softwares a utilizar durante la materia, tales como un IDE MPLab, así como el software que usaremos para realizar la carga de programas al microcontrolador, en el caso de este primer proyecto, al microcontrolador PIC16F788A.

Por otro lado, se realizó la programación basándose en nuevas instrucciones vistas previamente en clase, dentro de las cuales encontramos el uso y manejo de registros así como la configuración de nuevos puertos, tanto de salida como de entrada, para la interacción del microcontrolador con dispositivos externos, como lo es en este caso, la implementación de leds.

Por último, se retoman conceptos de materias anteriores, tales como circuitos eléctricos para realizar el alambrado del circuito propuesto durante la clase.

Hernández Jaimes Rogelio Yael

El microcontrolador PIC16F788A es una herramienta muy útil para conformar sistemas. Como se ha visto en la teoría, esta nos permite operar diferentes dispositivos externos y hacerlos operar en conjunto. Ahora bien, por sí mismo, el microcontrolador no tiene la capacidad de todo esto. Requiere ser programada y adicionalmente necesita de elementos que la hagan funcionar, como la alimentación, oscilador, entre otros. El sistema mínimo, precisamente provee de todo lo esencial para poder utilizar el microcontrolador y programarlo. En una parte, de todos los elementos eléctricos para que funcione adecuadamente. Por otra parte, incluye conexiones que permiten programar el microcontrolador desde una computadora, lo que facilita enormemente el trabajo. Si bien, este sistema mínimo puede hacer operaciones, no tiene forma visible de observar los resultados, para ello se necesitan periféricos y más componentes, sin embargo, precisamente por eso es el “Sistema mínimo”.