UNIVERSIDAD SIMÓN BOLÍVAR

DEPARTAMENTO DE ELECTRÓNICA Y CIRCUITOS

LABORATORIO DE MICROPROCESADORES EC-3074

INFORME - PRÁCTICA #3

Profesor

Mauricio Pérez

Estudiante

Giancarlo Torlone 20-10626

ÍNDICE

INTRODUCCIÓN

En la siguiente práctica de laboratorio se diseña una cerradura electrónica con el microcontrolador PIC16F877A que tiene un código de entrada de cuatro dígitos programados previamente (se puede tomar cuatro dígitos del carnet o de la cédula). Se debe conectar un teclado matricial que tenga los números del 0 al 9 más los símbolos de * y # directamente en el puerto B del microcontrolador (este puerto está diseñado expresamente para trabajar con teclados de este tipo). Adicionalmente el montaje debe tener un display de 7 segmentos conectado a cualquiera de los otros puertos del micro (el cual irá mostrando los números ingresados) y un par de leds, uno de color verde y otro de color rojo, junto con un zumbador. Estos componentes indicarán que la apertura de la puerta fue exitosa o hubo un error en la clave de acceso; tanto el led verde como el rojo deben encender por un periodo de 2 segundos para dar tiempo al usuario de abrir la puerta o para que escuche el zumbido y vuelva a ingresar la clave.

MARCO TEÓRICO

Teclado Matricial

Un Teclado Matricial es un simple arreglo de botones conectados en filas y columnas, de modo que se pueden leer varios botones con el mínimo número de pines requeridos. Los teclados matriciales son dispositivos que agrupan los pulsadores en filas y columnas formando una matriz. Pueden ser conectados a cualquier microcontrolador o son compatibles con Arduino. Sus aplicaciones van desde sistemas de seguridad, selección de menús o ingreso de datos en un sistema.

PIC16F877A

El PIC16F877A es un circuito integrado programable tipo FLASH reprogramable capaz de realizar y controlar tareas. El MCU cuenta con una RAM de 256 Bytes, frecuencia de trabajo de 20 MHz, empaquetado DIP-40 . Pertenece a la familia de microcontroladores PIC16.
El microcontrolador depende de una alimentación de al menos 5V y 0V en sus entradas de Vdd y Vss respectivamente para su operación, requiere de una señal de reloj que le indique la frecuencia de trabajo, está señal la introducimos a través de un oscilador de cristal de cuarzo, y una alimentación al pin MCLR, que es un pin de reset que activa al microcontrolador. El funcionamiento del microcontrolador está determinado por un programa almacenado en su memoria Flash ROM y puede programarse más de una vez para cambiar su estado y su comportamiento, lo que convierte al microcontrolador en una pieza esencial en el rápido desarrollo de aplicaciones electrónicas.
Algunas de sus aplicaciones son automatización y control de procesos, comunicaciones y red, electrónica de consumo, diseño embebido y desarrollo, multimedia, dispositivos portátiles, robótica, instrumentación o seguridad.

Algunas características del PIC16F877A

- 100.000 ciclos de borrado/escritura Enhanced Flash memoria del programa típica
- 1.000.000 de borrado/ciclo de escritura Datos EEPROM memoria típica
- Retención EEPROM de datos > 40 años
- Auto-reprogramable bajo control de software
- Programación serie en circuito(ICSP) a través de dos pines

- Programación serie de 5V in-circuit de un solo suministro
- Temporizador watchdog (WDT) con su propio RC en chip oscilador para un funcionamiento fiable
- Protección programable del código
- Ahorro de energía Modo de suspensión
- Opciones de oscilador seleccionables
- Depuración en circuito (ICD) a través de dos pines

Algunas instrucciones para los registros

ADDWF	Suma de W & F
ANDWF	Función AND de W & F
CLRF	Borrar un Registro
CLRW	Borra el registro de trabajo W
COMF	Complementa el Registro F
DECF	Decrementa F en 1
DECFSZ	Decrementa en 1 y salta si el resultado 0
INCF	Incrementa el registro F
INCFSZ	Incrementa en 1 y salta si el registro es 0
IORWF	Función OR de W & F
MOVF	Mover el registro F

RLF	Rota el registro F a la izquierda
RRF	Rota el registro F a la derecha
SUBWF	Resta F – W
SWAPF	Intercambio de F
XORWF	Función XOR de W & F
NOP	No operación
BCF	Borra un bit
BSF	Activa un bit
BTFSC	Verifica un bit y salta si es 0
BTFSS	Verifica un bit y salta si es 1
ANDLW	(W AND Literal)
CALL	Llamada a subrutina
CLRWTD	Borra el watchdog timer
GOTO	Salto incondicional
IORLW	(W OR Literal)
MOVLW	Carga un Valor al Registro W

RETURN	Regresa de una Subrutina
RETLW	Regresa de una Subrutina y carga el valor K en W
RETFIE	Regresa de la rutina de servicio
SLEEP	Entra en estado de reposo
XORLW	Realiza la función XOR entre W & K, el resultado se almacena en W
SUBLW	Resta L - W
MOVWF	Mover el valor del registro W al registro F

ANÁLISIS DE RESULTADOS

Oscilador

El oscilador del microcontrolador se configuró para que sea HS (High Speed) y para que trabaje a una frecuencia de 20 MHz.

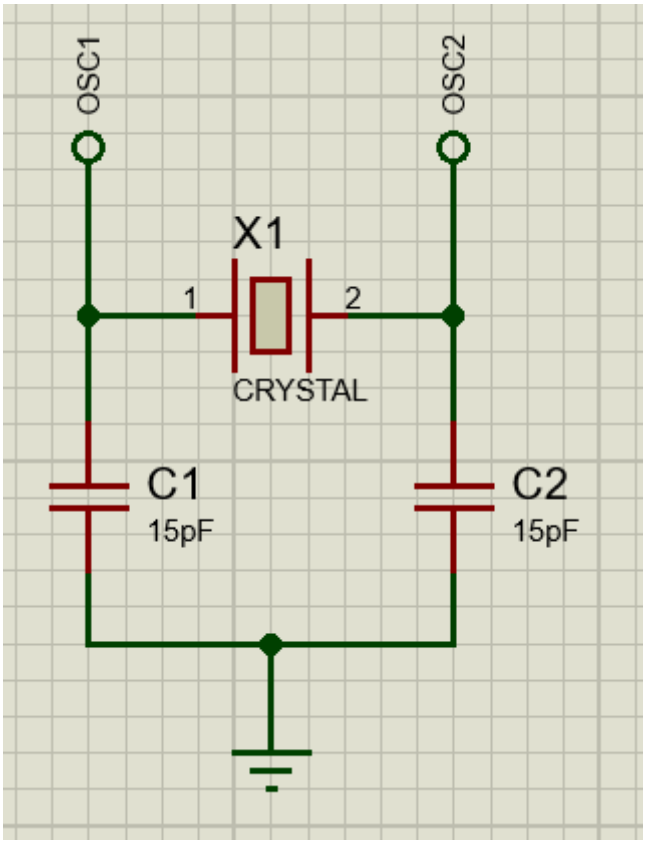

Figura 1. Conexiones para el oscilador

MCLR

Para el MCLR o reset se realizó la siguiente conexión en PULL UP.

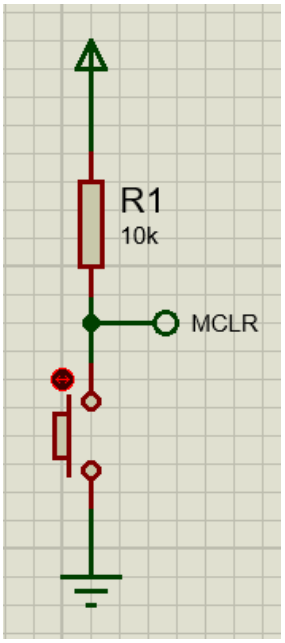

Figura 2. Conexión para el MCLR

Montaje Final

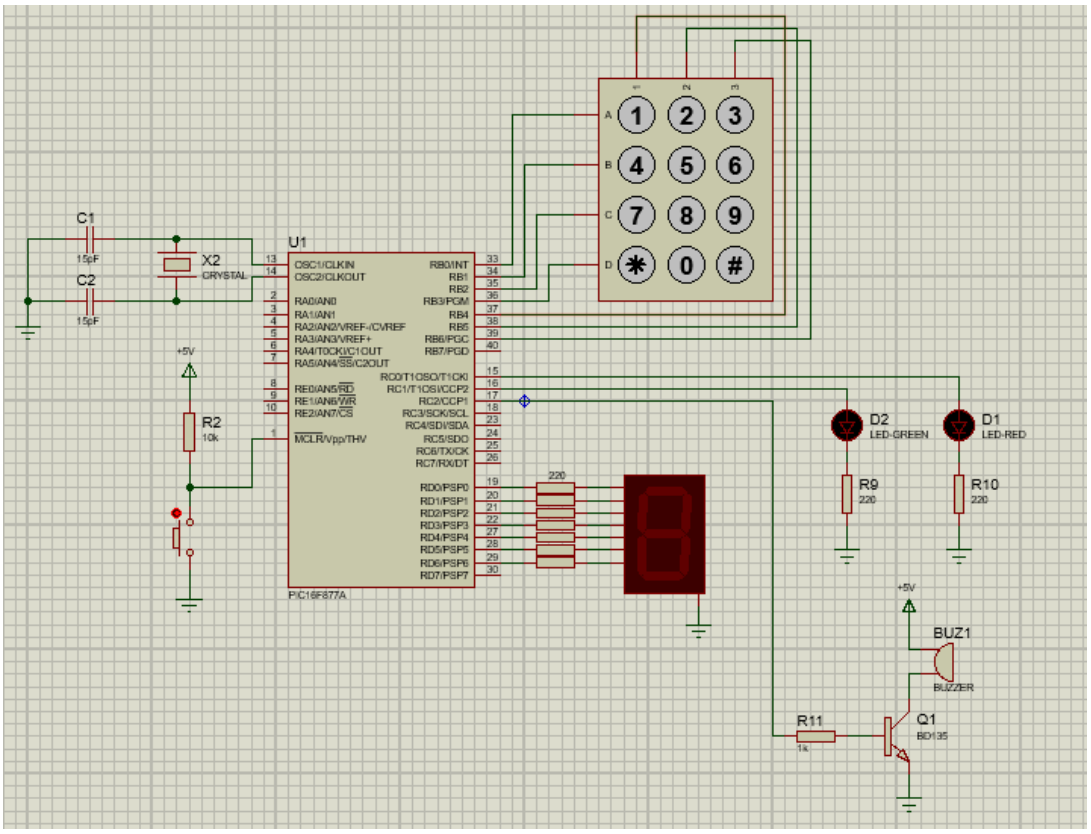

Figura 3. Montaje final del circuito

Código

Para facilitar el código, se hizo uso de librerías, las cuales nos permiten utilizar el teclado, los retardos y el display de 7 segmentos. También se hizo uso de macros para que el código sea más sencillo. Estas librerías y macros fueron obtenidas de la página https://rodrigocarita.com/tutorial/1, la cual cuenta con diversos tutoriales de microcontroladores PIC en assembler.

Las librerías fueron incluidas en las últimas líneas de código.

```
INCLUDE <teclado4x4.asm> ; libreria para teclado
INCLUDE <display.ASM> ; libreria display 7 segmentos
INCLUDE <RETARDOS.ASM> ; libreria para retardos

END
```

Se comienza definiendo las configuraciones del microcontrolador y las variables. También se incluye el archivo que contiene las macros.

```
#include "p16f877a.inc"
__CONFIG _FOSC_HS & _WDTE_OFF & _PWRTE_ON & _BOREN_OFF & _LVP_OFF & _CPD_OFF & _WRT_OFF & _CP_OFF
LIST P=16F877A
INCLUDE <mimacro.inc>
CBLOCK 0x20 ; bloque de memorias
teclapulsada ; Variable donde se guarda la 1ra tecla pulsada
teclapulsada2 ; Variable donde se guarda la 2da tecla pulsada
teclapulsada3 ; Variable donde se guarda la 3ra tecla pulsada
teclapulsada4 ; Variable donde se guarda la 4ta tecla pulsada
ENDC ; fin del bloque de memorias
ORG 0
```

Procedemos a definir el comportamiento de los puertos.
banco es una macro que permite ir al banco que se desea.
iniciar_teclado es una macro que define las configuraciones del teclado y permite inicializarlo utilizando la librería *teclado4x4*.

```
banco 1 ; ir al banco 1
CLRF TRISD ; puerto D como salida (display)
BCF TRISC, 0 ; puerto RC0 como salida (led rojo - acceso denegado)
BCF TRISC, 1 ; puerto RC1 como salida (led verde - acceso permitido)
BCF TRISC, 2 ; puerto RC2 como salida (buzzer)
banco 0 ; ir al banco 0
CLRF PORTD      ;Limpiamos puerto D (Apagamos el display)
BCF PORTC, 0 ;led rojo inicialmente apagado
BCF PORTC, 1 ;led verde inicialmente apagado
BCF PORTC, 2 ;buzzer inicialmente apagado
iniciar_teclado ;Iniciamos el teclado matricial 4x4 (configuraciones)
```

A continuación se encuentra el programa principal, el cual se encarga de almacenar los 4 dígitos presionados y compararlos con la clave seleccionada **0626** (últimos 4 dígitos del carnet de la universidad 2010626). En caso de ser errónea, se llama a la subrutina DENEGADO, la cual encenderá el led rojo y activará el buzzer que actuará como alarma; en caso de ser correcta, se activa el led verde indicando que la clave ingresada es la correcta y se abriría la puerta.

leertecla es una macro que utiliza la librería *teclado4x4* y permite leer la tecla presionada por el usuario.
milisegundo es una macro que utiliza la librería de retardos para generar un delay con los milisegundos que se deseen.

```
PROGRAMA

leertecla               ;leemos 1ra tecla
MOVWF teclapulsada      ;Guardamos la tecla pulsada en el registro
registrocdisplay teclapulsada ;Escribimos la teclapulsada en el display (catodo comun)
milisegundo .255 ; delay antirebote

leertecla               ;leemos 2da tecla
MOVWF teclapulsada2      ;Guardamos la tecla pulsada en el registro
registrocdisplay teclapulsada2 ;Escribimos la teclapulsada en el display (catodo comun)
milisegundo .255 ; delay antirebote

leertecla               ;leemos 3ra tecla
MOVWF teclapulsada3      ;Guardamos la tecla pulsada en el registro
registrocdisplay teclapulsada3 ;Escribimos la teclapulsada en el display (catodo comun)
milisegundo .255 ; delay antirebote

leertecla               ;leemos 4ta tecla
MOVWF teclapulsada4      ;Guardamos la tecla pulsada en el registro
registrocdisplay teclapulsada4 ;Escribimos la teclapulsada en el display (catodo comun)
milisegundo .255 ; delay antirebote
```

csni también es una macro, la cual compara un registro con un valor y se va a la rutina indicada si no son iguales. Aquí se compara si los 4 dígitos ingresados son iguales a la clave programada **0626.** Si no son iguales se llama a la subrutina DENEGADO.

```
; ---------------------- Sección de la contraseña: 0626 --------------------
csni teclapulsada,.0, DENEGADO ; Si al consultar la contraseña, 1 digito falla
csni teclapulsada2,.6, DENEGADO ; automaticamente se irá a la subrutina
csni teclapulsada3,.2, DENEGADO ; llamada denegado
csni teclapulsada4,.6, DENEGADO
```

En caso de que sí sean iguales, se llama a la subrutina PERMITIDO.

```
; ---------------------- Secuencia de entrada permitida -------------------
; Si se ha llegado aquí es porque la contraseña es correcta.
CALL PERMITIDO
GOTO PROGRAMA           ;finaliza el programa
```

Finalmente definimos las subrutinas DENEGADO y PERMITIDO.

La subrutina DENEGADO es la que se encarga de encender el led rojo y la alarma, ya que se introdujo una clave errónea.
segundo (al igual que milisegundo) es una macro que permite establecer un delay con los segundos deseados.

```
DENEGADO ; aquí empieza la subrutina en caso de que la contraseña sea erronea
BSF PORTC, 0 ; encendemos led rojo
BSF PORTC, 2 ; encendemos buzzer - alarma
segundo .2 ; delay de 2 segundos
BCF PORTC, 0 ; apagamos led rojo
BCF PORTC, 2 ; apagamos buzzer - alarma
CLRF PORTD
GOTO PROGRAMA ; reiniciamos el programa
```

La subrutina PERMITIDO es la que se encarga de encender el led verde y abriría la puerta, pues se introdujo la clave correcta.

```
PERMITIDO
BSF PORTC, 1 ; encendemos led verde
segundo .2 ; delay de 2 segundos
BCF PORTC, 1 ; apagamos led verde
CLRF PORTD
GOTO PROGRAMA ; reiniciamos el programa
```

Limitaciones con el simulador PROTEUS

El programa cumple su función: cuando se introduce la clave correcta **0626** se enciende el led verde y cuando se introduce una clave incorrecta se enciende el led rojo y el buzzer.

Sin embargo, es necesario destacar que la simulación presenta fallas al momento de introducir la clave, pues no maneja muy bien el delay de antirebote. En ocasiones cuando se presiona una tecla, el delay de antirebote no funciona correctamente y es posible que dicha tecla se pulse dos veces seguidas, ocasionando que se detecte una clave errónea. Por lo que hay que intentarlo varias veces.

El circuito en físico no debería presentar esta falla pues 255 ms (delay para antirrebote) es bastante tiempo.

Código completo

```
#include "p16f877a.inc"
__CONFIG _FOSC_HS & _WDTE_OFF & _PWRTE_ON & _BOREN_OFF & _LVP_OFF & _CPD_OFF & _WRT_OFF & _CP_OFF
LIST P=16F877A
INCLUDE <mimacro.inc>
CBLOCK 0x20 ; bloque de memorias
teclapulsada ; Variable donde se guarda la 1ra tecla pulsada
teclapulsada2 ; Variable donde se guarda la 2da tecla pulsada
teclapulsada3 ; Variable donde se guarda la 3ra tecla pulsada
teclapulsada4 ; Variable donde se guarda la 4ta tecla pulsada
ENDC ; fin del bloque de memorias
ORG 0

banco 1 ; ir al banco 1
CLRF TRISD ; puerto D como salida (display)
BCF TRISC, 0 ; puerto RC0 como salida (led rojo - acceso denegado)
BCF TRISC, 1 ; puerto RC1 como salida (led verde - acceso permitido)
BCF TRISC, 2 ; puerto RC2 como salida (buzzer)
banco 0 ; ir al banco 0
CLRF PORTD      ;Limpiamos puerto D (Apagamos el display)
BCF PORTC, 0 ;led rojo inicialmente apagado
BCF PORTC, 1 ;led verde inicialmente apagado
BCF PORTC, 2 ;buzzer inicialmente apagado
iniciar_teclado ;Iniciamos el teclado matricial 4x4 (configuraciones)

PROGRAMA

leertecla               ;leemos 1ra tecla
MOVWF teclapulsada      ;Guardamos la tecla pulsada en el registro
registrocdisplay teclapulsada ;Escribimos la teclapulsada en el display (catodo comun)
milisegundo .255 ; delay antirebote
```

```
leertecla               ;leemos 2da tecla
MOVWF teclapulsada2      ;Guardamos la tecla pulsada en el registro
registrocdisplay teclapulsada2 ;Escribimos la teclapulsada en el display (catodo comun)
milisegundo .255 ; delay antirebote

leertecla               ;leemos 3ra tecla
MOVWF teclapulsada3      ;Guardamos la tecla pulsada en el registro
registrocdisplay teclapulsada3 ;Escribimos la teclapulsada en el display (catodo comun)
milisegundo .255 ; delay antirebote

leertecla               ;leemos 4ta tecla
MOVWF teclapulsada4      ;Guardamos la tecla pulsada en el registro
registrocdisplay teclapulsada4 ;Escribimos la teclapulsada en el display (catodo comun)
milisegundo .255 ; delay antirebote

; ---------------------- Sección de la contraseña: 0626 --------------------
csni teclapulsada,.0, DENEGADO ; Si al consultar la contraseña, 1 digito falla
csni teclapulsada2,.6, DENEGADO ; automaticamente se irá a la subrutina
csni teclapulsada3,.2, DENEGADO ; llamada denegado
csni teclapulsada4,.6, DENEGADO

; ---------------------- Secuencia de entrada permitida ------------------
; Si se ha llegado aquí es porque la contraseña es correcta.
CALL PERMITIDO
GOTO PROGRAMA           ;finaliza el programa
```

```
DENEGADO ; aquí empieza la subrutina en caso de que la contraseña sea erronea
BSF PORTC, 0 ; encendemos led rojo
BSF PORTC, 2 ; encendemos buzzer - alarma
segundo .2 ; delay de 2 segundos
BCF PORTC, 0 ; apagamos led rojo
BCF PORTC, 2 ; apagamos buzzer - alarma
CLRF PORTD
GOTO PROGRAMA ; reiniciamos el programa

PERMITIDO
BSF PORTC, 1 ; encendemos led verde
segundo .2 ; delay de 2 segundos
BCF PORTC, 1 ; apagamos led verde
CLRF PORTD
GOTO PROGRAMA ; reiniciamos el programa

INCLUDE <teclado4x4.asm> ; libreria para teclado
INCLUDE <display.ASM> ; libreria display 7 segmentos
INCLUDE <RETARDOS.ASM> ; libreria para retardos

END
```