Instituto Tecnológico y de Estudios

Superiores de Occidente – ITESO

ITESO

Universidad Jesuita
de Guadalajara

Materia: Fundamentos de microprocesadores y microcontroladores

Profesor: Álvaro Gutiérrez Arce

Práctica 3

Fecha: 25 de noviembre de 2021

Tema: Práctica 3: Display LCD

Autores: Miriam Guadalupe Malta Reyes

Rodrigo Zamora Davalas

Lista de contenidos

Desarrollo técnico

Los LEDs son útiles para indicar que un programa está corriendo, está conectado o está esperando el estatus de alguna señal externa; sin embargo, no pueden mostrar todos los caracteres ASCII. Un display LCD, a diferencia de los LEDs puede mostrar números y todos los caracteres ASCII junto con algunos caracteres especiales. Además, la interfaz entre el display y el microcontrolador es sencilla ya que el monitoreo de todos los caracteres se hace de forma automática en la circuitería de control interna de los módulos LCD.

Objetivos

- Ejercitar el uso de puertos, subrutinas, interrupciones y puerto serial en un microcontrolador.
- Aprender el manejo de un teclado matricial y una pantalla de cristal líquido (LCD).
- Desarrollar una interfaz entre el microcontrolador y un dispositivo con comunicación serial.
- Abrir la perspectiva respecto de los periféricos que se le pueden conectar a un microcontrolador, con el fin de aumentar las capacidades del alumno para utilizarlo en una aplicación real.
- Fortalecer las bases teóricas y prácticas para el desarrollo de interfaces con el microcontrolador, con el fin de fomentar la creatividad y motivar el generar ideas para el desarrollo del proyecto final del curso con orientación a una aplicación real.

Desarrollo

Implementar un programa que permita desplegar en el LCD la información que se introduzca a través del teclado. Se deben filtrar los rebotes del teclado para evitar múltiples lecturas de la misma tecla y evitar que el display parpadeé.

Cada vez que se introduzca un nuevo carácter debe colocarse a la derecha del anterior. Una vez que se llena la primera línea debe saltar al inicio de la segunda. Cuando la segunda línea se llene, debe borrar la pantalla y regresar al inicio de la primera.

Además, se debe agregar un botón que genere una interrupción que permita introducir caracteres tipo ASCII. Al dejar presionado el botón, se podrán introducir dos dígitos en hexadecimal a través del teclado y se mostrara el carácter ASCII correspondiente en el

display LCD. Por ejemplo, si no se presiona este botón y se oprime la tecla 4 enseguida se oprime la tecla 5, en el display LCD se mostrará un 4 y a la derecha un 5. Ahora bien, al presionar este botón, y oprimir la tecla 4 enseguida se oprime la tecla 5, en el display LCD se mostrará el carácter E; en código ASCII el carácter E tiene un código igual a 45 en hexadecimal.

Finalmente se debe agregar un segundo botón que genere una interrupción para que la información que se está desplegando en la pantalla, se transmita vía el puerto serial del micro a una computadora personal conectada al sistema por medio de un módulo de bluetooth inalámbrico.

Para el armado de la alarma utilizamos:

- 1 - AT89S52
- 2 - protoboards
- Alambre
- 2 - Pushbutton
- 1 - Resistencia de 8.2k ohms
- 1 - Resistencia de 330 ohms
- 1 - Capacitor de 10uF
- 1 - Cristal de Quarzo de 11 MHz
- 2 - Capacitores de 33p
- 1 - Display LCD
- 1 - Teclado matricial
- 1 - Módulo bluetooth hc05

Esquemático

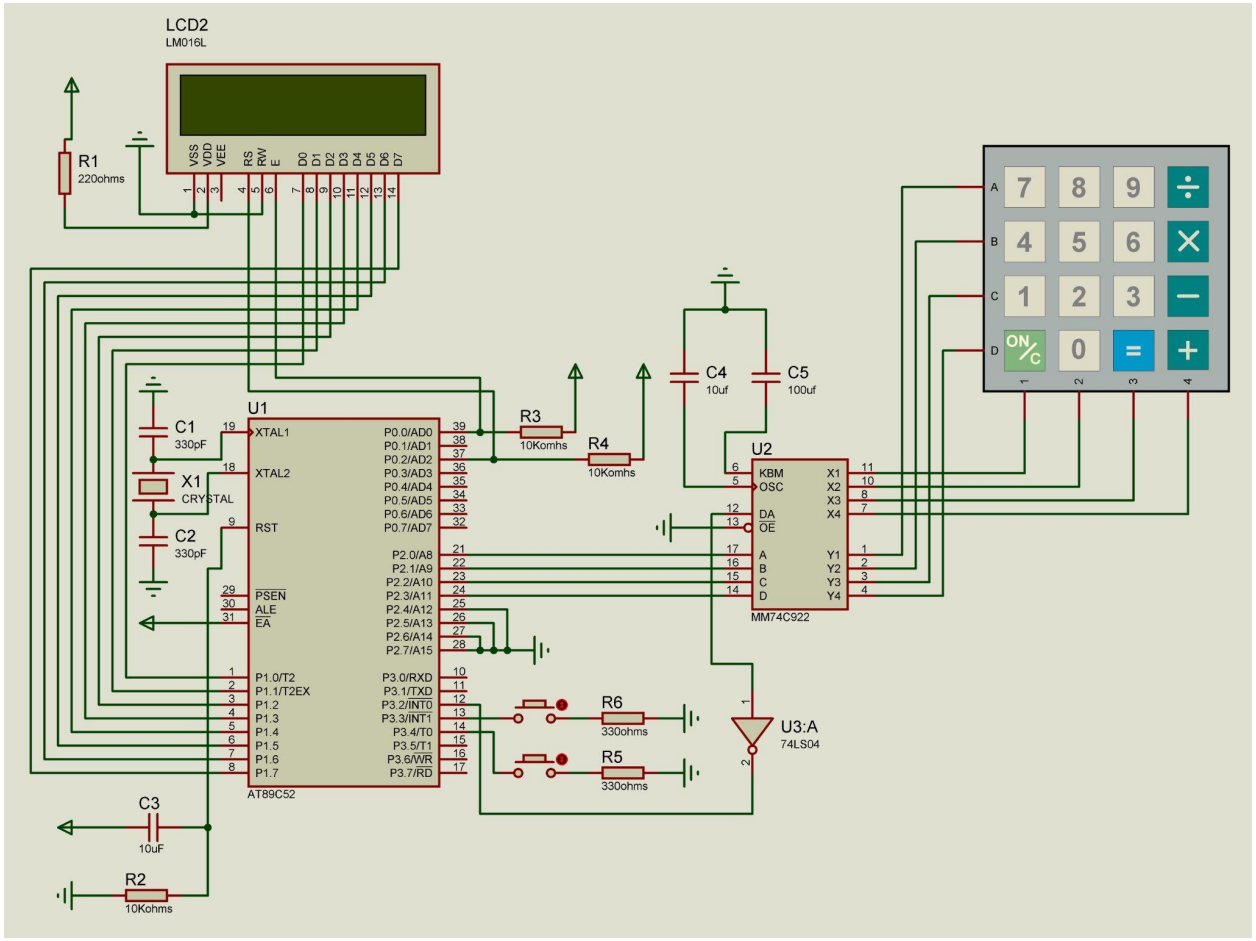
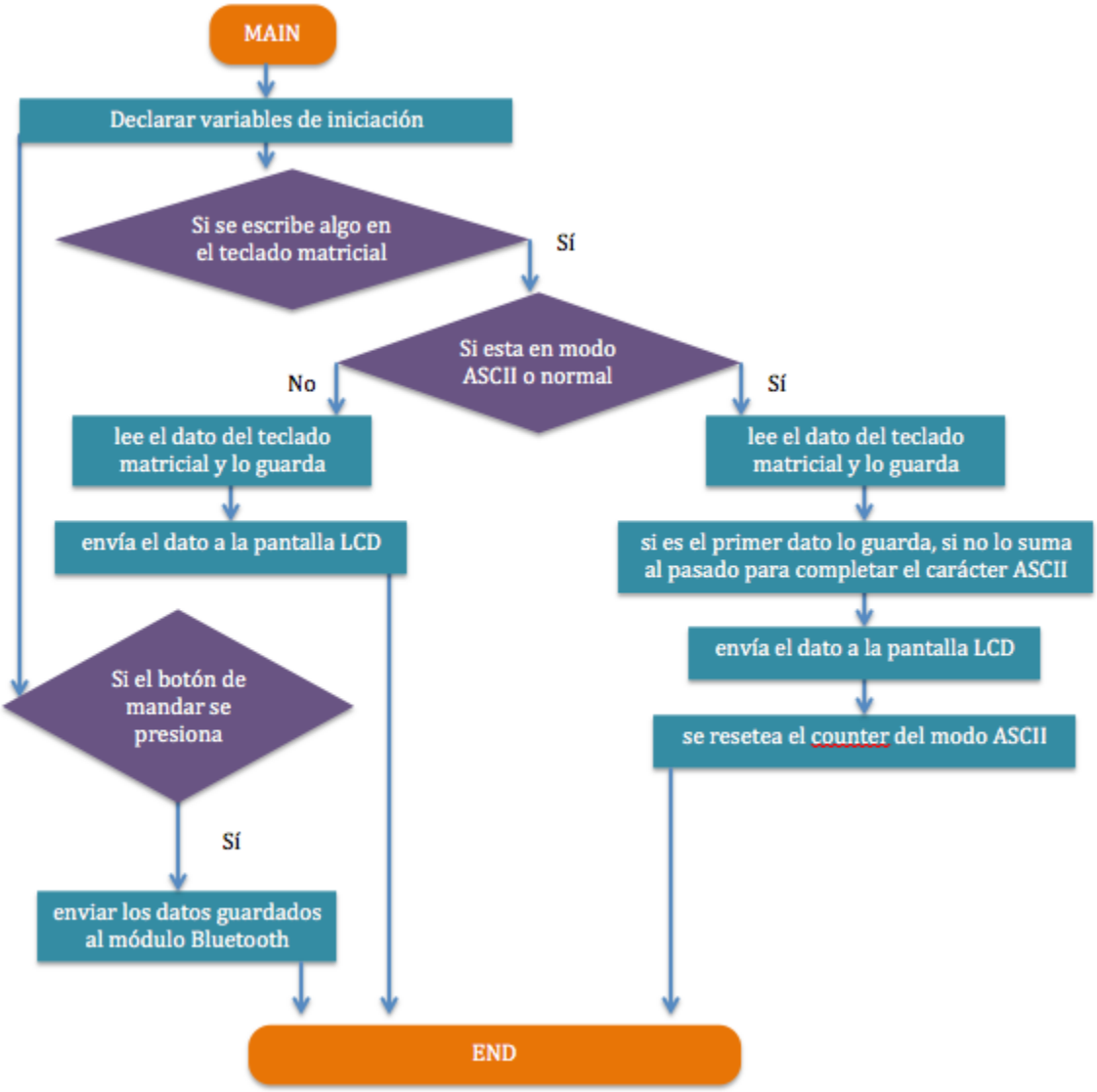


Diagrama de flujo



Código del programa con comentarios

```
ORG 0000H
JMP MAIN

ORG 0003H
        JMP KEYBOARD_INPUT                      // Interrupción INT0

ORG 0013H
        JMP KEYBOARD_INPUT_2            // Interrupción INT1

ORG 00040H
MAIN:
        ACALL RESET_MEMORY
        ACALL INIT_LCD_DISPLAY
        ACALL INIT_INTERRUPTIONS
        ACALL INIT_SERIAL_TIMER

        MAIN_LOOP:
                JNB P3.4, $
                INC R6
                SETB P0.3
                JB P3.4, $

        CJNE R6, #00H, $                        // Esperar a que keyboardCounter sea reseteado
por la interrupción, para poder recibir otro input
        JMP main_loop



KEYBOARD_INPUT:                                 // Interrupción 1, leer el dato del teclado
amtricial, guardarlo en memoria RAM y enviarlo a la pantalla LCD. Si keyboardInput está en 2,
estamos en modo ASCII
        CJNE R6, #02H, KEYBOARD_INPUT_1
        MOV A, R4
        ADD A, P2
        MOV R4, A
        MOV R6, #00H
        ACALL LCD_SEND_DATA
        CLR P0.3
        RETI

        KEYBOARD_INPUT_1:                       // Si no es dos, revisamos si es 1 para
guardar el input en el acumulador
```

```
            CJNE R6, #01H, KEYBOARD_INPUT_NORMAL
            MOV A, P2
            SWAP A
            MOV R4, A
            INC R6
        RETI

        KEYBOARD_INPUT_NORMAL:          // Si no es 1 ni 2, guardamos el input del
teclado y lo enviamos
            MOV A, P2
            ACALL CHANGE_TO_ASCII
            MOV R4, A
            ACALL LCD_SEND_DATA
        RETI


        KEYBOARD_INPUT_2:               // Enviar los datos guardados en la memoria
ram por serial
            MOV R1, #30H

        MEMORY_LOOP:
            MOV A, @R1
            ACALL SEND_ACC_SERIAL
            INC R1
            CJNE R1, #50H, MEMORY_LOOP
        RETI

RESET_MEMORY:
        MOV R4, #00H
        MOV R2, #00H
        MOV R3, #01H
        MOV R0, #030H
        MOV R6, #00H
        SETB P3.4
        CLR P0.3
        CLR P0.2
        CLR P0.1
        CLR P0.0
        CLR TI

        MOV R1, #30H

        MEMORY_RESET_LOOP:
            MOV @R1, #00H
```

```
            INC R1
            CJNE R1, #50H, MEMORY_RESET_LOOP
        RET

RESET_RAM_AND_LCD:
        MOV R1, #30H
        MOV R4, #01H
        ACALL LCD_SEND_INSTRUCTION

        ACALL LCD_CHANGE_ROW_TO_1

        RESET_RAM_AND_LCD_LOOP:
            MOV @R1, #00H
            INC R1
            CJNE R1, #50H, RESET_RAM_AND_LCD_LOOP
        RET

INIT_SERIAL_TIMER:                          // Iniciar el timer 1 para poder transmitir a
9600 baudios
        MOV SCON, #01000000B
        MOV TMOD, #00100000B
        MOV TH1, #0FDH
        MOV TL1, #(-3)
        SETB TR1
        RET

INIT_LCD_DISPLAY:
        MOV R4, #38H
        ACALL LCD_SEND_INSTRUCTION
        ACALL LONG_DELAY

        MOV R4, #38H
        ACALL LCD_SEND_INSTRUCTION
        ACALL LONG_DELAY

        MOV R4, #38H
        ACALL LCD_SEND_INSTRUCTION
        ACALL LONG_DELAY

        MOV R4, #01H
        ACALL LCD_SEND_INSTRUCTION
        ACALL LONG_DELAY

        MOV R4, #0FH
```

```
        ACALL LCD_SEND_INSTRUCTION
        ACALL LONG_DELAY
        RET

LCD_SEND_DATA:
        SETB P0.2
        MOV P1, R4
        SETB P0.0
        NOP
        CLR P0.0

        ACALL SHORT_DELAY
        ACALL STORE_R4

        RET

LCD_SEND_INSTRUCTION:
        CLR P0.2
        MOV P1, R4
        SETB P0.0
        NOP
        CLR P0.0
        RET

INIT_INTERRUPTIONS:
        MOV IE, #10000101B
        SETB IT0
        SETB IT1
        RET

STORE_R4:
        MOV A, R4
        MOV @R0, A
        INC R0

        CJNE R0, #40H, NO_CHANGE_1
        ACALL LCD_CHANGE_ROW_TO_2

        NO_CHANGE_1:
                CJNE R0, #50H, NO_CHANGE_2
                ACALL RESET_RAM_AND_LCD
                MOV R0, #30H
                ACALL RESET_RAM_AND_LCD
```

```
        NO_CHANGE_2:
                RET

LCD_CHANGE_ROW_TO_2:
        MOV R4, #0C0H
        ACALL LCD_SEND_INSTRUCTION
        RET

LCD_CHANGE_ROW_TO_1:
        MOV R4, #80H
        ACALL LCD_SEND_INSTRUCTION
        RET


SEND_ACC_SERIAL:
        MOV SBUF, A
        JNB TI, $
        CLR TI
        RET

CHANGE_TO_ASCII:
        MOV R5, A
        ASCII_1:
                CJNE A, #00H, ASCII_2
                MOV A, R5
                ADD A, #30H
                RET
        ASCII_2:
                CJNE A, #0AH, ASCII_3
                MOV A, R5
                ADD A, #37H
                RET
        ASCII_3:
                DEC A
                SJMP ASCII_1

LONG_DELAY:
        MOV R3, #15H

SHORT_DELAY:
        MOV R2, #0FFH
        DJNZ R2, $
        DJNZ R3, SHORT_DELAY
        MOV R3, #1D
```

```
        RET

END
```

Conclusiones individuales

Miriam Guadalupe Malta Reyes

Con esta práctica me ayudó a entender cómo se hace para poder pasar datos al display lcd, como usar un teclado matricial, como se puede pasar de teclado matricial a display lcd y como mandar estos datos a través del módulo bluetooth. Especialmente la utilidad tanto del display lcd como del módulo bluetooth, ya que con estos puedes checar que es lo que está pasando y si sí se está escribiendo lo que se desea que se escriba pues tienes una respuesta física de que está bien o mal. Me ayudó a terminar de entender muchos temas vistos en clase y una aplicación visible de un microcontrolador.

Rodrigo Zamora Dávalos

Esta práctica me ayudó a entender como usar la comunicación serial y poder darle como un uso más real al microcontrolador, al poder ver los datos transmitidos en la computadora. Tuvimos demasiados problemas con el cableado, llegando a cambiar varios componentes, y aún así no podíamos hacerlo funcionar, pero en la simulación que hacíamos si servía, usando el mismo código y esquemático. No supimos porque era.

Referencias

Manish K Patel (2014). The 8051 Microcontroller Based Embedded Systems. McGraw Hill Education