Reporte Laboratorio #5 Microcontroladores

Emanuel Lascurain A01552126
Victor Cavazos A01177055

Procedimientos

Usamos el ejemplo dado por el profesor para detectar el botón presionado en nuestro teclado de 4x4, a través de energizar nuestras cuatro filas en orden, y detectar en cada una si había un 0, ya que significa la acción de nuestra resistencia “pull up”, y en base a esto, la función regresa un valor del 0 al 15 dependiendo del botón presionado.
El siguiente paso es declarar nuestros 16 cases y darle el valor binario requerido para nuestros displays de 7 segmentos
Ahora para nuestra función encargada de desplegar en los displays, lo requerido es que muestre el valor que acaba de ser presionado, y después lo guarde para pasarlo al siguiente display de la izquierda cuando se presione otro botón y muestre el nuevo valor.
Para esto, la solución a la cual llegamos, fue ir cambiando el valor de nuestro código hexadecimal a través de ANDs y ORs para finalmente desplegarlo.

```
void send_to_disp(uint32_t disp_word){
        LATB = (char) 0xFF;
        char Lock = disp_word;
        char x;
        char y;
        char w;

        if (disp_word != 'x'){
        disp_word = 0x000000FF & disp_word;
        disp_word = disp_word | (x>>24);
        x=y;
        disp_word = disp_word | (y>>16);
        y=w;
        disp_word = disp_word | (w>>8);
        w=Lock;
        uint8_t num_disp = disp_word;
        LATD = char_to_seg(num_disp);
        }
        else ();
        __delay_ms(SWEEP_FREQ);
    }
```

Primero tenemos 4 variables y a la primera “Lock” le damos el valor del botón siendo presionado, la que debe mostrar.
Luego ponemos un if para que solo llame a la función en caso de tener un botón presionado, y cuando no sea el caso, el valor es ‘x’ por eso el AND NOT.
Le damos el valor disp_word de 000000FF & disp_word para que se guarde el valor en los primeros 8 bits.
Después usamos un OR para guardar ambos valores, en este caso a si mismo y el de nuestra variable x con un shift de 24 bits, en los primeros casos este valor es 0, pero después tendrá valores anteriores, por eso después de esto x=y.
Después repetimos la acción con nuestras siguiente variables y teniendo shifts diferentes para cambiar los bits correspondientes.
Finalmente desplegamos.
Así conseguimos guardar los valores ya presionados en nuestras variables x, y, w y Lock y los colocamos en orden en nuestro código de 32 bits para finalmente desplegarlo.

Resultados

Con la elaboración de esta práctica pudimos comprender y aplicar el uso de los teclados matriciales, así como de los displays de 7 segmentos. El resultado principal de esta práctica es una interacción funcional entre nuestro teclado de 4x4, que está formado por un arreglo matricial de 16 botones pulsador es, donde cada uno tiene asignado un valor alfanumérico el cuál al ser presionado nos muestra en el display dicho valor, los display están formados por una serie de 7 diodos emisores de luz, y con la programación podemos generar los números y letras dependiendo de qué presione el usuario, posteriormente el valor se recorre hacia la izquierda.

Conclusiones

Victor: Logre comprender de manera más sencilla el uso de códigos en forma de muchos bits, binario y hexadecimal y su gran utilidad a través de operadores lógicos y como podemos simplificar el uso de componentes electrónicos puede llegar a parecer que la solución es meter mas pines o tener un código muy largo y confuso.

Emanuel:
Se consiguió hacer uso efectivo de los puertos GPIO de la placa Curiosity para interactuar con el mundo real, pudiendo introducir datos dados por un usuario mediante el uso de un teclado matricial, así como representar valores de salida en un conjunto de LEDs agrupados en forma de display. Considero que estos periféricos son ampliamente utilizados en la industria, por lo que es fundamental entendamos su funcionamiento y aplicación.