UNIVERSIDAD DE SEVILLA

UNIVERSIDAD D SEVILLA

INGENIERÍA MECATRÓNICA

Sistemas Electrónicos

Trabajo realizado por:

Pedro López Japón

Raúl Antonio Pérez Dorantes

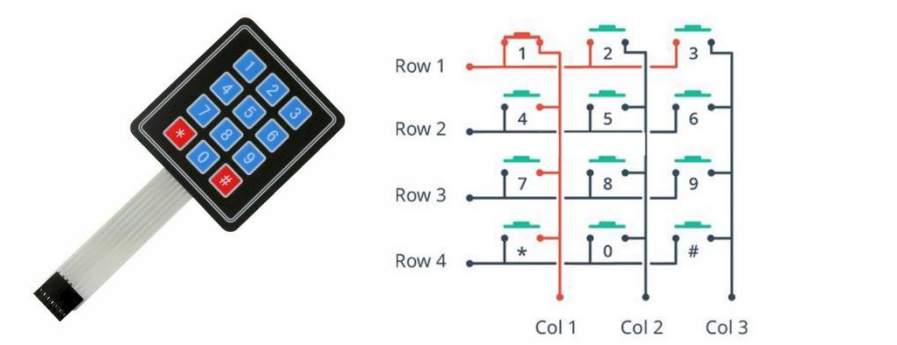
Índice de contenido

Las cajas fuertes tienen el trabajo de resguardar un objeto de valor no necesariamente monetario, esta son empleadas por individuos o bien, empresas. El proyecto consiste en una caja fuerte virtual, la cual tiene conectada un teclado matricial para la entrada de valores, un motor controlado por PWM para simular la apertura y una pantalla LCD como interfaz de usuario.

Marco teórico

Teclado matricial

Un teclado matricial es un dispositivo que agrupa varios pulsadores y permite controlarlos empleando un número de conductores inferior al que necesitaríamos usarlos de forma individual. Estos dispositivos agrupan los pulsadores en filas y columnas formando una matriz.



Servomotor

Son actuadores rotativos que permiten un control preciso en términos de posición angular, aceleración y velocidad, capacidades que un motor normal no tiene. Sin embargo, los servomotores no son en realidad una clase especifica de motor, sino una combinación de piezas específicas, que incluyen un motor de corriente continua o alterna, y son adecuados para su uso en un sistema de control de buble cerrado.

El modelo en particular que se empela, es el **SG90**, un servomotor pequeño que puede rotar aproximadamente 180 grados. La ventaja que ofrece este modelo en particular, es la facilidad para controlarlo, puesto a que podemos emplear una librería o programar manualmente el ciclo de trabajo de la señal PWM para ajustar su posición, para efecto del proyecto, será programado manualmente.

PWM

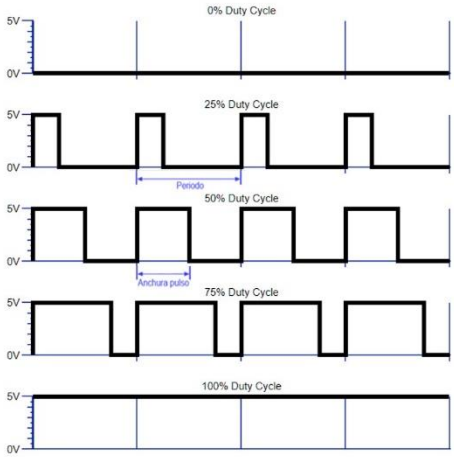
Por sus siglas en inglés *"Pulse Width Modulation"* (Modulación por ancho de pulso) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica, ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el periodo. Expresando matemáticamente:

$$D = \frac{\tau}{T}$$

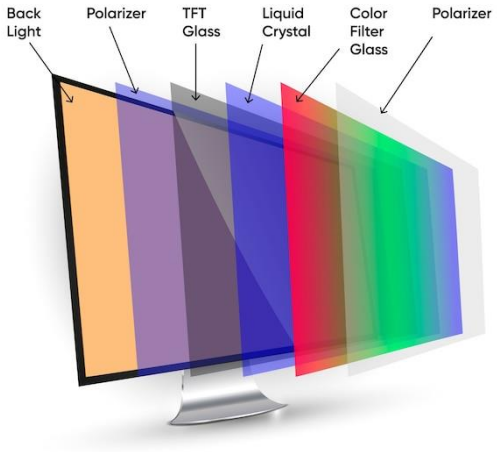
Donde:

- D es el ciclo de trabajo.
- τ es el tiempo en que la función es positiva (ancho de pulso).
- T es el periodo de la función.



Pantalla LCD

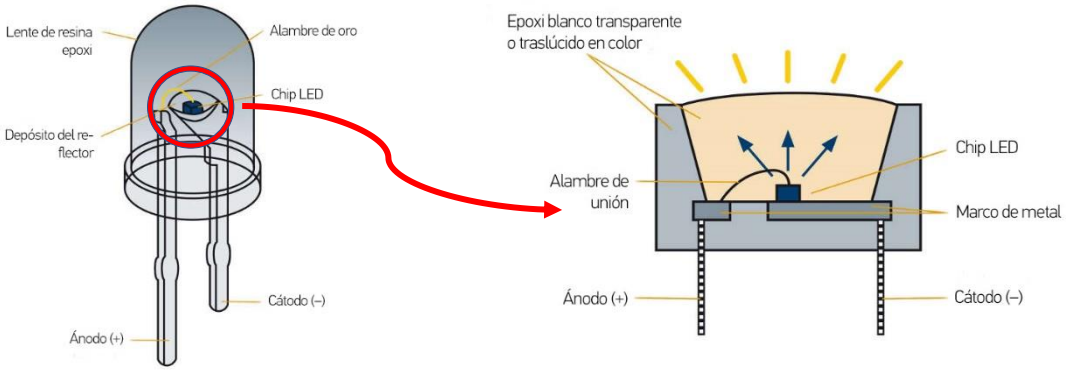
Liquid Crystal Display o Pantallas de cristal líquido son pantallas delgadas y planas formadas por un número de pixeles en color o monocromos colocados delante de una fuente de luz o reflectora. A menudo se utiliza en dispositivos electrónicos de pilas, ya que utiliza cantidades muy pequeñas de energía eléctrica.



La pantalla LCD para el proyecto es la incluida en el *Educational Boosterpack II* la cual lleva por modelo CFAF128128B-0145T de la marca Crystalfontz. Esta pantalla posee una resolución de 128x128 pixeles, y se emplean librerías para facilitar su control.

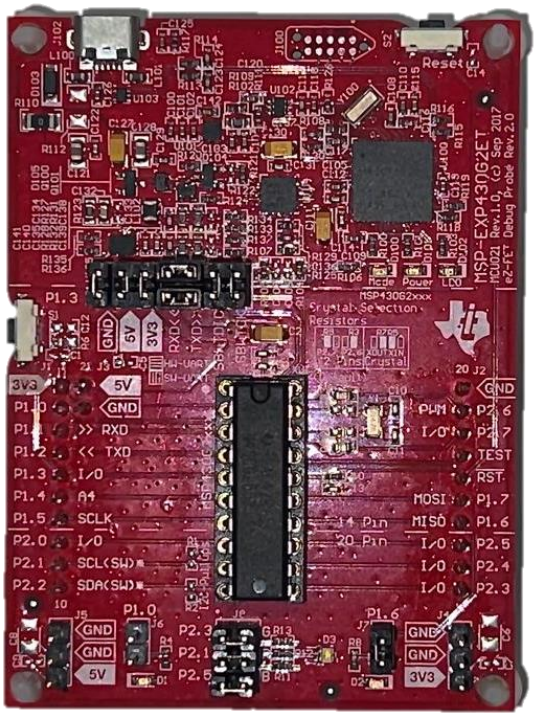
LED

También conocido como diodo emisor de luz, es un diodo de unión p-n, que emite luz al estar activado. Si se aplica una tensión adecuada a los terminales, los electrones se recombinan con los nuevos en la región de la unión p-n del dispositivo, liberando energía en forma de fotones.



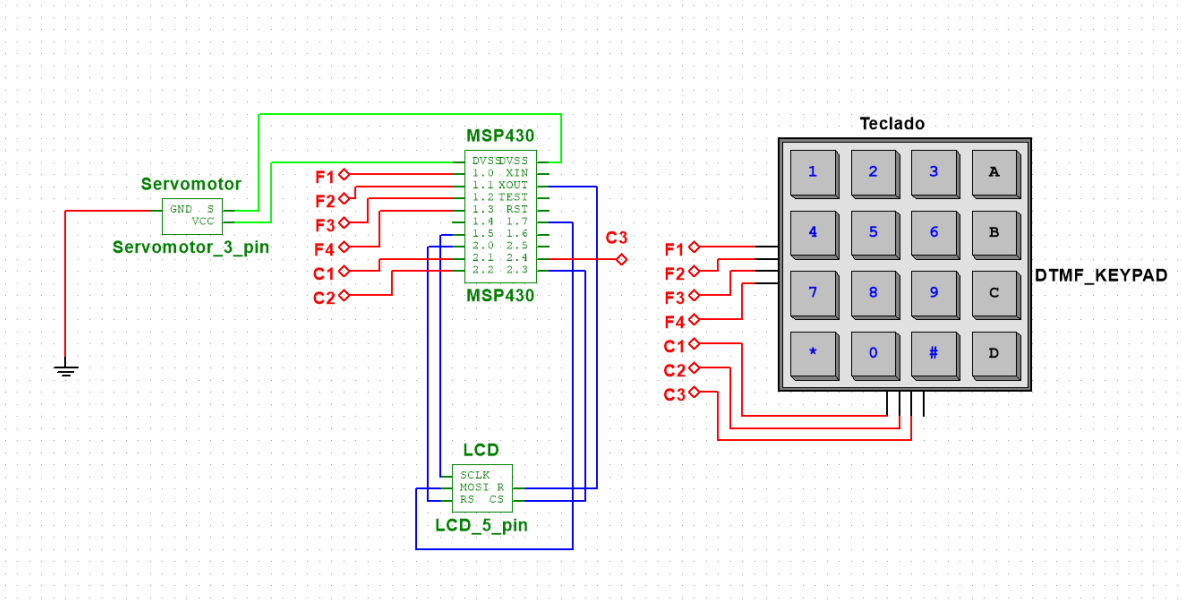
Puertos del Microcontrolador

Para poder implementar el sistema, es necesario realizar conexiones físicas para cada uno de los componentes, y estos componentes a su vez pueden tener requerimientos específicos para su control. Para poder controlar todos los componentes se empleará el microcontrolador MSP430G2553 de Texas Instruments. El cual posee un total de 20 pines, cada uno de estos pines posee una función, sin embargo, al estar en una placa de desarrollo tenemos un diagrama más fácil de comprender a primera vista.



Puerto		Descripción
Microcontrolador	LaunchPad	
Boosterpack MKII		
3.3v	J2.1	Alimentación a 3.3 volts.
GND	J3.2	Tierra física del circuito.
Servomotor		
2.6	J2.2	Señal PWM.
Pantalla LCD		
1.5	J1.7	Reloj síncrono.
1.7	J2.6	Master Output, Slave Input.
2.0	J1.8	Register Select
2.3	J2.10	Chip Select
2.7	J2.3	Reset
Teclado Matricial		
2.1	N/A	Columna 1
2.2	N/A	Columna 2
2.4	N/A	Columna 3
1.0	N/A	Fila 1
1.1	N/A	Fila 2
1.2	N/A	Fila 3
1.3	N/A	Fila 4
LEDS		
2.5	N/A	Rojo
1.4	N/A	Azul
1.6	N/A	Verde

Diagrama eléctrico



Tomando como referencia anterior tabla, las conexiones realizadas en cada uno de los pines dan como resultado el esquema de conexiones de la imagen anterior.

Como nota al pie, cabe destacar que los pines, como puede ser el caso del XOUT, corresponden al reloj, sin embargo, estos pueden usados como pines de propósito general. De igual manera, tenemos que el último pin de las columnas no se encuentra conectado, esto no es un error, puesto a que el teclado matricial implementado es un 4x3(4 filas, 3 columnas), por lo tanto, no es necesario realizar la conexión de la cuarta columna.

Código

El funcionamiento del programa gira en torno a una máquina de estados, la cual alterna su función dependiendo de la opción que el usuario seleccione.

Lo primero a comprender es el teclado matricial, recordemos que es esencialmente un array de botones los cuales están distribuidos en 3 columnas y 4 filas y cada una de las teclas se ve definida como la posición perpendicular de estas; por ejemplo, si nosotros queremos pulsar el botón número 5, en el array tenemos que suposición sería columna 2 y fila 2.

Se define todo lo anterior con variables:

- *keymap_char:* esta variable será un array de 4x3, en cada una de las posiciones d esta variable se colocará el carácter correspondiente de la tecla siendo presionada.
- *fil_pins:* la variable tendrá una longitud de 4x1, en las cuales cada posición corresponde a los pines a utilizar.
- *col_pins:* su longitud será de 3x1, e igualmente a la anterior, cada posición corresponde a los pines a usar.

Una vez definido lo anterior, se debe de crear la función encargada de leer cual es la tecla pulsada, para ello se definen como salida los pines correspondientes a las filas, mientras que los pines de las columnas serán entradas. De igual manera las filas tendrán un valor lógico 0 al momento de ser inicializadas mientras que las columnas tendrán el valor lógico 1 con una resistencia pull-up. La resistencia tipo-pull-up es indispensable, puesto a que al momento de que una tecla es presionada, el valor de la columna correspondiente pasa a ser 0, de esta manera podemos identificar qué botón está siendo presionado en ese instante.

Para identificar la posición de la tecla correspondiente al array *keymap_char*, es necesario generar un *for* para verificar el estado actual de los botones y al momento de que uno sea presionado, los valores de las filas pasarán a ser 1.

Como los valores de las teclas son ahora un 1, y un botón está siendo presionado, uno de los valores de las filas será un 0 debido a que su señal va a tierra. Para verificar cada una de ellas, se emplea nuevamente un *for* (tendrá la variable *fil* que será empleada para identificar la posición en el array).

Posteriormente, tenemos que alguna de las columnas tiene un botón siendo presionado, por lo tanto, nuevamente algún valor será 0 y esto se verifica de igual manera con un ciclo *for* (el cual tendrá la variable *col* que será empleada para identificar la posición en el array).

Para finalizar, empleamos las funciones de escritura en la pantalla, asignando *keymap_char* los valores de las variables *fil* y *col* para identificar el carácter.

En el código se ve de la siguiente manera:

```
char keymap_char[4][3] = {'1', '2', '3',
                          '4', '5', '6',
                          '7', '8', '9',
                          '*', '0', '#'};

P1DIR |= (BIT0 | BIT1 | BIT2 | BIT3);
P2DIR &=~ (BIT1 | BIT2 | BIT4);
P2REN |= (BIT1 | BIT2 | BIT4);
P2OUT |= (BIT1 | BIT2 | BIT4);
P1OUT &=~ (BIT0 | BIT1 | BIT2 | BIT3);

char fil_pins[4] = {BIT0, BIT1, BIT2, BIT3};
char col_pins[3] = {BIT1,BIT2,BIT4};
```

Donde tenemos:

- *keymap_char* se colocan cada uno de los caracteres.
- Los pines 1.0 a 1.3 son los pines de las filas, los cuales están definidos como salida y su valor inicial es 0.

- Los pines 2.1, 2.2, 2.4 corresponden a los pines de las columnas, están definidos como entrada y su valor inicial es 1.
- Se generan las variables *fil_pins* y *col_pins* cuyas longitudes serán 4 y 3 respectivamente y llevarán dentro de cada posición uno de los pines correspondientes para poder alternar cada una de las posiciones en un ciclo *for* posterior, esto optimiza el programa.

Como nota al pie tenemos igualmente otras variables:

```
unsigned char  fil=0,col=0,col2=0,
char cadena[100];
```

- *fil*: Empleada dentro de un *for,* permite alternar entre las posiciones del vector.
- *col:* Nos permite saber si un botón está siendo presionado.
- *col2:* Nuevamente, empleada dentro de un *for,* permite alternar entre las posiciones del vector.
- *cadena:* Será empleada en la función para imprimir caracteres en la pantalla.

Definido lo anterior, se coloca en el ciclo *while* lo siguiente:

```
while(1)
{
    LPM0;
    P1OUT&=~fil_pins[0];
    P1OUT&=~fil_pins[1];
    P1OUT&=~fil_pins[2];
    P1OUT&=~fil_pins[3];
    //si se ha pulsado una tecla
    for (col=0;col<3;col++)
    {
        if(!(P2IN&col_pins[col]))
        {
            P1OUT|=fil_pins[0];
            P1OUT|=fil_pins[1];
            P1OUT|=fil_pins[2];
            P1OUT|=fil_pins[3];

            //barrido por filas
            for (fil=0;fil<4;fil++)
            {
                (P1OUT&=~fil_pins[fil]);

                //barrido en columnas
                for (col2=0;col2<3;col2++)
                {
                    if(!(P2IN&col_pins[col2]))
                    {
                        Graphics_clearDisplay(&g_sContext);
                        Graphics_setBackgroundColor(&g_sContext, GRAPHICS_COLOR_BLACK);
                        Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_WHITE);
                        Graphics_drawString(&g_sContext,"Tecla: ", 15, 20, 60, TRANSPARENT_TEXT);
                        sprintf(cadena,"%c",keymap_char[fil][col]);
                        Graphics_drawString(&g_sContext,cadena, 15, 80, 60, OPAQUE_TEXT);

                    }
                    P2OUT|=col_pins[col2];
                }
                P1OUT|=fil_pins[fil];
            }
            P2OUT|=col_pins[col];
        }
    }
}
```

Nuevamente hacemos que los valores de las filas sean un 0, esto es para que su valor vuelva a ser el por defecto después de cada iteración.

El primer ciclo *for* es el encargado de alternar entre cada una de las columnas, y dentro se tiene una condición en la cual si alguno de los pines del puerto 1 está siendo presionado, hace que los valores de salida de las filas pasen a ser un 1.

Ahora se realiza una comprobación en cada una de las filas para saber cuál de los valores es un 0, recordemos que esto se debe a que su señal está a tierra. Cuando algunas de las señales detectadas sea 0, entonces se procede a realizar la comprobación de las columnas.

Nuevamente, si las alguna de las columnas tiene ahora un 0, entonces asignamos *col* y *char* a las posiciones de *keymap_char* para así obtener un carácter. Las demás funciones dentro de esta parte, corresponden a la pantalla.

Al momento de Salir de cada uno de los *if*, regresamos a su valor original cada una de las salidas.

Lo siguiente es la máquina de estados que debe de alternar entre cada uno de los siguientes estados posibles:

- Definir contraseña: este es el *setup* de nuestra caja fuerte, el usuario debe de ingresar una contraseña para que esta sea usada posteriormente.
- Cerrado: en este estado la caja fuerte está cerrada.
- Introducir contraseña: Contiene la función del teclado matricial, el usuario debe de introducir la contraseña para poder abrir la caja, en caso de que sea incorrecta, la caja fuerte regresa al estado cerrado
- Abierto: la caja fuerte está abierta. El usuario puede cerrar la caja fuerte y si así lo desea puede cambiar la contraseña.

Cabe destacar que la función del teclado ha sido ligeramente modificada con la intención de optimizar el código.

El código principal contiene la configuración de los pines, los relojes y las librerías a emplear, recordemos que estamos usando la pantalla LCD y existen librerías que han sido otorgadas y explicadas en clase. Los relojes poseen la función de, además fungir como parte del control de la pantalla, controlar el servomotor por PWM.

```
#include <msp430.h>
#include "grlib.h"
#include "Crystalfontz128x128_ST7735.h"
#include "HAL_MSP430G2_Crystalfontz128x128_ST7735.h"
#include <stdio.h>
```

Incluimos las librerías.

```
void espera(int T)
{
    int i;
    for(i=0;i<T; i++)
    {
        __delay_cycles(910);    //Función espera con el nro de ciclos
    }
}

void Set_Clk(char VEL){
    BCSCTL2 = SELM_0 | DIVM_0 | DIVS_0;
    switch(VEL){
    case 1:
        if (CALBC1_1MHZ != 0xFF) {
            DCOCTL = 0x00;
            BCSCTL1 = CALBC1_1MHZ;      /* Set DCO to 1MHz */
            DCOCTL = CALDCO_1MHZ;
        }
        break;
    case 8:

        if (CALBC1_8MHZ != 0xFF) {
            __delay_cycles(100000);
            DCOCTL = 0x00;
            BCSCTL1 = CALBC1_8MHZ;      /* Set DCO to 8MHz */
            DCOCTL = CALDCO_8MHZ;
        }
        break;
    case 12:
        if (CALBC1_12MHZ != 0xFF) {
            __delay_cycles(100000);
            DCOCTL = 0x00;
            BCSCTL1 = CALBC1_12MHZ;     /* Set DCO to 12MHz */
            DCOCTL = CALDCO_12MHZ;
        }
        break;
    case 16:
        if (CALBC1_16MHZ != 0xFF) {
            __delay_cycles(100000);
            DCOCTL = 0x00;
            BCSCTL1 = CALBC1_16MHZ;     /* Set DCO to 16MHz */
            DCOCTL = CALDCO_16MHZ;
        }
        break;
    default:
        if (CALBC1_1MHZ != 0xFF) {
            DCOCTL = 0x00;
            BCSCTL1 = CALBC1_1MHZ;      /* Set DCO to 1MHz */
            DCOCTL = CALDCO_1MHZ;
        }
        break;

    }
    BCSCTL1 |= XT2OFF | DIVA_0;
    BCSCTL3 = XT2S_0 | LFXT1S_2 | XCAP_1;
}
```

Configuramos los relojes para la pantalla LCD.

```
char keymap_char[4][3] = {'1', '2', '3',
                          '4', '5', '6',
                          '7', '8', '9',
                          '*', '0', '#'};

Graphics_Context g_sContext;
enum {cerrado, abierto, introducir_contraseña,definir_contraseña};
volatile unsigned char  estado=definir_contraseña;
unsigned char  j1=0,x=0,digito=0,fil=0,col=0,col2=0,abrir=1,cerrar=0,emergencia=0;
unsigned int correct=0;
char fil_pins[4] = {BIT0, BIT1, BIT2, BIT3};
char col_pins[3] = {BIT1,BIT2,BIT4};

char cadena[40];
```

Definimos todas las variables a utilizar.

- **keymap_char:** un array de 4 filas y 3 columnas en la cuales se encuentran posicionados cada uno de nuestros **dígitos**.
- **enum:** es la máscara para los estados.
- **estado:** la variable que permite cambiar entre cada uno de los posibles estados, inicializamos está en “definir contraseña” para realizar un “setup” inicial.
- **j1:** Estará dentro de un ciclo *for* que nos permitirá realizar una animación.
- **x:** similar a la anterior variable, permite realizar una animación.
- **digito:** es la variable que nos permitirá obtener un número o dígito del teclado matricial.
- **fil:** variable dentro de un ciclo *for* la cual permite reconocer la fila siendo presionada.
- **col:** variable dentro de un ciclo *for* que permite identificar si una tecla está siendo presionada.
- **col2:** variable dentro de un ciclo *for* que permite identificar la columna siendo presionada.
- **abrir:** es una variable que controla al servomotor.
- **cerrar:** similar a la anterior, controla al servomotor.
- **emergencia:** esta variable permite realizar el cambio de contraseña.
- **fil_pins:** contiene los pines de los puertos de las filas, facilita su cambio.
- **col_pins:** similar a la anterior, contiene los puertos de las columnas.

Definido lo anterior, lo siguiente es el contenido del *main*.

```
int main(void)
{
    //contrx: contraseña que se intruduce / segx:codigo de seguridad de la puerta
    char contr1=0,contr2=0,contr3=0,contr4=0,seg1=0,seg2=0,seg3=0,seg4=0;


    WDTCTL = WDTPW | WDTHOLD; //Stop watchdog timer

    Set_Clk(16); //configuracion del reloj

    Crystalfontz128x128_Init(); //inicializacion de la pantalla del BP

    //inicializacion de puertos y direcciones usadas
    P1DIR |= (BIT0 | BIT1 | BIT2 | BIT3 | BIT4 | BIT6); //filas del teclado como salidas  y bits de los leds
    P2DIR &=~ (BIT1 | BIT2 | BIT4);//columnas del teclado como entradas
    P2DIR |= BIT5; //pin led salida
    P2REN |= (BIT1 | BIT2 | BIT4);//habilito resistencia en las entradas
    P2OUT |= (BIT1 | BIT2 | BIT4);//resistencias de pullup
    P2OUT &=~ BIT5;//led apagado
    P1OUT &=~ (BIT0 | BIT1 | BIT2 | BIT3 | BIT4 | BIT6);//inicializo salidas a cero y leds apagados

    //inicializacion del bit de PWM
    P2DIR|=BIT6;
    P2SEL = BIT6;
    P2SEL2 =0;

    /*  TA1: CON SMCLK*/
    TA1CTL=TASSEL_2|ID_3| MC_1;
    TA1CCR0=1999;       //periodo=20000: 10ms
    TA1CCTL0=CCIE;   //CCIE=1

    //configuracion timer A0 : servo
    TA0CCTL0=CCIE;      //CCIE=1
    TA0CCTL1=OUTMOD_7;      //OUTMOD=7
    TA0CTL=TASSEL_2| ID_3 |MC_1;        //SMCLK, DIV=8, UP
    TA0CCR0=39999;      //periodo=40.000= 20ms
    TA0CCR1=3000;       //Ton inicial: 1.5ms %

    //configuracion de la pantalla del BP
    Crystalfontz128x128_SetOrientation(LCD_ORIENTATION_UP);
    Graphics_initContext(&g_sContext, &g_sCrystalfontz128x128);
    Graphics_clearDisplay(&g_sContext);
    Graphics_setBackgroundColor(&g_sContext, GRAPHICS_COLOR_BLACK);
    Graphics_setFont(&g_sContext, &g_sFontCmss14);

    Graphics_Rectangle rectangulo = {50,30,80,80}; //puerta cerrada

    __bis_SR_register(GIE);
```

En esta parte se encuentra toda la configuración principal del programa, cabe destacar que aquí se han agregado dos variables:

- **contrx**: la cual es la contraseña siendo ingresada, posteriormente comparada con **segx**.
- **segx:** es el código de seguridad de la puerta.

```
while(1)
{
    LPM0; //entramos en bajo consumo

    switch(estado)
    {
    case cerrado:

        emergencia=0; //variable que hace que cuando la puerta este abierta, se pase a la configuracion de cambiar codigo de seguridad. eso pasa cuando emergencia=1.
        cerrar=0; // para controlar el servo cuando la puerta se este cerrando
        contr1=0; //reiniciamos las variables de la contraseña intruducida
        contr2=0;
        contr3=0;
        contr4=0;
        //pinto la puerta cerrada
        Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_BROWN);
        Graphics_fillRectangle(&g_sContext,&rectangulo);
        Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_WHITE);
        Graphics_drawLine(&g_sContext, 49,30,49,80);
        Graphics_drawLine(&g_sContext, 81,30,81,80);
        Graphics_drawLine(&g_sContext, 49,30,81,30);
        Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_BLACK);
        Graphics_drawCircle(&g_sContext, 55,55,2);
        Graphics_fillCircle(&g_sContext, 55,55,2);
        //escribo en pantalla
        Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_WHITE);
        Graphics_drawString(&g_sContext,"Door closed", 11, 10, 12, OPAQUE_TEXT);
        Graphics_drawString(&g_sContext,"*: insert password", 18, 10, 90, OPAQUE_TEXT);

        //para insertar contraseña pulso asterisco
        P1OUT&=~fil_pins[3];
        if(!(P2IN&col_pins[0]))
        {
            Graphics_clearDisplay(&g_sContext);
            estado=introducir_contraseña;
        }
        break;
```

Entramos al estado de bajo consumo y al despertarse el micro entramos a la máquina de estados. En esta parte hacemos que la contraseña ingresada por el usuario sean todas 0, pintamos una puerta y escribimos en pantalla. Un punto muy importante es que si el usuario pulsa la tecla asterisco (*), entonces el estado cambia a *introducir_contraseña*.

```
case abierto://se abre la puerta
    if(abrir==1)//para el servo
    {
        //escribo en pantalla
        Graphics_clearDisplay(&g_sContext);
        Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_WHITE);
        Graphics_drawString(&g_sContext,"Opening door", 12,20, 10, OPAQUE_TEXT);
        //pinto la puerta cerrada para poder abrirla
        Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_BROWN);
        Graphics_fillRectangle(&g_sContext,&rectangulo);
        Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_WHITE);
        Graphics_drawLine(&g_sContext, 49,30,49,80);
        Graphics_drawLine(&g_sContext, 81,30,81,80);
        Graphics_drawLine(&g_sContext, 49,30,81,30);
        Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_BLACK);
        Graphics_fillCircle(&g_sContext, 55,55,2);

        //pinto la puerta abriendose
        Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_WHITE);
        Graphics_drawLine(&g_sContext, 49,30,49,80);
        Graphics_drawLine(&g_sContext, 81,30,81,80);
        Graphics_drawLine(&g_sContext, 49,30,81,30);
        Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_BLACK);
        for(j1=50;j1<66;j1++){
            Graphics_drawLine(&g_sContext,j1,80,j1,31);
            Graphics_drawLine(&g_sContext,50,130-j1,130-j1,130-j1);
            Graphics_fillCircle(&g_sContext, j1+2,50,1);
            espera(5000);
        }
        abrir=0;
    }
```

Este es el estado abierto, el cual cambia la interfaz del usuario a una puerta siendo abierta.

```
else if(abrir==0)
{
    //puerta abierta
    P1OUT&=~fil_pins[3];
    Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_WHITE);
    Graphics_drawString(&g_sContext,"#: change password", 18, 5, 90, OPAQUE_TEXT);
    Graphics_drawString(&g_sContext,"Door opened  ", 12,20,10, OPAQUE_TEXT);
    Graphics_drawString(&g_sContext,"Time left", 10,5, 110, OPAQUE_TEXT);
    espera(500);
    for(x=10;x>0;x--)//cuenta atras para el cierre automatico de la puerta
    {
        espera(18000);
        if(!(P2IN&col_pins[2]))//si se pusa # se pasa a cambiar la clave de seguridad
        {
            x=1;
            emergencia=1;
        }
        Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_BLACK);
        Graphics_fillCircle(&g_sContext, 83,115,9);
        Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_WHITE);
        sprintf(cadena,"%d",x);
        Graphics_drawString(&g_sContext,cadena, 5, 77, 110, OPAQUE_TEXT);
        if(x==1)//fin de la cuenta atras
        {
            cerrar=1;
            if(emergencia==1)////si se pusa # se pasa a cambiar la clave de seguridad
            {
                Graphics_clearDisplay(&g_sContext);
                espera(1000);
                estado=definir_contraseña;
            }
            else
            {
                //se empieza a cerrar la puerta despues de la cuenta atras
                Graphics_drawString(&g_sContext,"Closing door", 12,20,10, OPAQUE_TEXT);
                for(j1=65;j1>49;j1--){

                    Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_BROWN);
                    Graphics_drawLine(&g_sContext,j1,31,j1,130-j1);
                    Graphics_drawLine(&g_sContext,80,130-j1,j1,130-j1);
                    Graphics_fillCircle(&g_sContext, j1+4,50,1);
                    Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_BLACK);
                    Graphics_fillCircle(&g_sContext, j1+2,50,1);

                    espera(5000);
                }
                Graphics_clearDisplay(&g_sContext);
                estado=cerrado;
            }
        }
    }
}
break;
```

Es parte del estado *abierto,* reconoce el momento en que la animación de la puerta a finalizado y le otorga al usuario la capacidad de cambiar la contraseña, para efectos del proyecto, se ha cambiado la capacidad de cerrar la puerta de manera manual a automática.

Cuando el tiempo se ha agotado entonces la interfaz vuelve a cambiar y el usuario ahora debe de esperar a que la puerta se cierre para nuevamente elegir si desea ingresar la contraseña.

```
case introducir_contraseña:
    espera(10000);
    emergencia=0;
    Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_WHITE);
    Graphics_drawString(&g_sContext,"Entering password...", 25, 8, 30, OPAQUE_TEXT);
    //codigo de uso del teclado
    if(digito<4)//para que solo se puedan pulsar 4 numeros para la contraseña
    {
        P1OUT&=~fil_pins[0];//ponemos las filas (salidas) a 0 para poder reconocer si se ha pulsado alguna tecla
        P1OUT&=~fil_pins[1];
        P1OUT&=~fil_pins[2];
        P1OUT&=~fil_pins[3];
        //si se ha pulsado una tecla
        for (col=0;col<3;col++)
        {
            if(!(P2IN&col_pins[col]))//si el valor de alguna columna pasa a cero es que se ha pulsado una tecla
            {
                P1OUT|=fil_pins[0];//ponemos las filas a 1 para ir poniendolas a 0 una a una para ver en que fila se ha pulsado la tecla
                P1OUT|=fil_pins[1];
                P1OUT|=fil_pins[2];
                P1OUT|=fil_pins[3];

                //barrido por filas
                for (fil=0;fil<4;fil++)
                {
                    (P1OUT&=~fil_pins[fil]);//ponemos una a una las filas a cero

                    //barrido en columnas
                    for (col2=0;col2<3;col2++)
                    {
                        if(!(P2IN&col_pins[col2]))//si la columna esta a cero cuando este a cero la fila, entonces esa [fila][columna] es la tecla pulsada
                        {
                            Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_WHITE);
                            sprintf(cadena,"%c",keymap_char[fil][col]);//pinto la tecla pulsada
                            Graphics_drawString(&g_sContext,cadena, 15,digito*30+10, 80, OPAQUE_TEXT);
                            espera(10000);
                            if(digito==0)//guardo cada tecla pulsada en una variable de contraseña
                                contr1=keymap_char[fil][col];
                            else if(digito==1)
                                contr2=keymap_char[fil][col];
                            else if(digito==2)
                                contr3=keymap_char[fil][col];
                            else if(digito==3)
                                contr4=keymap_char[fil][col];

                            digito++;
                        }
                    }
                }
            }
        }
    }
```

Es parte es fundamental, puesto a que este estado contiene la función de ingresar la contraseña.

```
else
{
    P1OUT&=~fil_pins[3];//si se pulsa # se pasa a comprobar si la contraseña coincide con la clave de seguridad
    if(!(P2IN&col_pins[2]))
    {
        Graphics_clearDisplay(&g_sContext);
        if((contr1==seg1) && (contr2==seg2) && (contr3==seg3) && (contr4==seg4))//si esto ocurre la contraseña es buena y se abre la puerta
        {
            Graphics_drawString(&g_sContext,"Correct password", 16, 5, 50, OPAQUE_TEXT);
            espera(10000);
            Graphics_clearDisplay(&g_sContext);
            abrir=1;
            estado=abierto;
            digito=0;
        }
        else//la contraseña no se la buena y se pasa al estado de cerrado de nuevo
        {
            Graphics_drawString(&g_sContext,"Incorrect password", 18, 5, 50, OPAQUE_TEXT);
            espera(10000);
            Graphics_clearDisplay(&g_sContext);
            estado=cerrado;
            digito=0;
        }
    }
}

break;
```

El estado sigue siendo el mismo, si el usuario pulsa la tecla gato(#) entonces se realiza la comprobación de la contraseña, si la contraseña es incorrecta entonces la caja fuerte regresa al estado *cerrado*.

```
case definir_contraseña://se crea la clave de seguridad de la puerta

    espera(10000);
    emergencia=0;
    Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_WHITE);
    Graphics_drawString(&g_sContext,"Define password", 15, 8, 30, OPAQUE_TEXT);
    //codigo de funcionamiento del teclado igual que antes
    if(digito<4)
    {
        P1OUT&=~fil_pins[0];
        P1OUT&=~fil_pins[1];
        P1OUT&=~fil_pins[2];
        P1OUT&=~fil_pins[3];
        //si se ha pulsado una tecla
        for (col=0;col<3;col++)
        {
            if(!(P2IN&col_pins[col]))
            {
                P1OUT|=fil_pins[0];
                P1OUT|=fil_pins[1];
                P1OUT|=fil_pins[2];
                P1OUT|=fil_pins[3];

                //barrido por filas
                for (fil=0;fil<4;fil++)
                {
                    (P1OUT&=~fil_pins[fil]);

                    //barrido en columnas
                    for (col2=0;col2<3;col2++)
                    {
                        if(!(P2IN&col_pins[col2]))
                        {
                            Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_WHITE);
                            sprintf(cadena,"%c",keymap_char[fil][col]);
                            Graphics_drawString(&g_sContext,cadena, 15,digito*30+10, 80, OPAQUE_TEXT);
                            espera(10000);
                            if(digito==0)
                                seg1=keymap_char[fil][col];
                            else if(digito==1)
                                seg2=keymap_char[fil][col];
                            else if(digito==2)
                                seg3=keymap_char[fil][col];
                            else if(digito==3)
                                seg4=keymap_char[fil][col];

                            digito++;

                        }
                    }
                }
            }
        }
    }
```

El estado *definir_contraseña* realiza la función indicada en su nombre, se encarga de definir una nueva contraseña.

```
else//en este caso cuando se pulse # se pasa a cerrado directamente
{
    P1OUT&=~fil_pins[3];
    if(!(P2IN&col_pins[2]))
    {
        digito=0;
        Graphics_clearDisplay(&g_sContext);
        espera(10000);
        estado = cerrado;
    }
}
break;
```

Y si el usuario pulsa la tecla gato (#) entonces la caja fuerte pasa al estado *cerrado*.

```
#pragma vector=TIMER1_A0_VECTOR
__interrupt void Interrupcion_T1(void)
{
    LPM0_EXIT;
    //encendido y apagado de los leds
    if (estado==abierto)//puerta abierta se enciende el verde
    {
        P2OUT &=~ BIT5;
        P1OUT &=~ BIT4;
        P1OUT |= BIT6;
    }
    if(estado==cerrado)//puerta cerrada se enciende el rojo
    {
        P2OUT |= BIT5;
        P1OUT &=~ BIT4;
        P1OUT &=~ BIT6;
    }
    if (estado==introducir_contraseña || estado==definir_contraseña)//cuando se esta usando el teclado se enciende el azul
    {
        P2OUT &=~ BIT5;
        P1OUT |= BIT4;
        P1OUT &=~ BIT6;
    }
}

#pragma vector=TIMER0_A0_VECTOR
__interrupt void TIMER0_A0_ISR_HOOK(void)
{
    LPM0_EXIT;
    //posicion del servo con PWM
    if(abrir==1)//el servo pasa a su estado maximo para abrirse
    {
        TA0CCR1+=20;
        if(TA0CCR1>=5050)
            TA0CCR1=5050;
    }
    if(cerrar==1)//el servo pasa a su estado minimo para cerrarse
    {
        TA0CCR1-=20;
        if(TA0CCR1<=1150)
            TA0CCR1=1150;
    }
    if(estado==definir_contraseña)//mantenemos el servo cerrado cuando se usa el teclado
        TA0CCR1=1150;
    if(estado==introducir_contraseña)
            TA0CCR1=1150;
    if(estado==cerrado)
        TA0CCR1=1150;
}
```

Para finalizar tenemos las interrupciones, en el *TIMER 1* se encuentra el encendido y apagado de los leds dependiendo el estado en el que se encuentre la máquina de estados.

En el *TIMER 0* se encuentra el control del servomotor dependiendo de la máquina de estados y las variables *abrir* y *cerrar.*

Resultados

Asdasd

Bibliografía

- Llamas, L. (2019, 18 julio). *Usar un teclado matricial con Arduino*. Luis Llamas. https://www.luisllamas.es/arduino-teclado-matricial/
- aula21. (2022, 16 diciembre). *Qué es un Servomotor y para qué sirve*. aula21 | Formación para la Industria. https://www.cursosaula21.com/que-es-un-servomotor/
- colaboradores de Wikipedia. (2022, 31 octubre). *Modulación por ancho de pulsos*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_ancho_de_pulsos
- colaboradores de Wikipedia. (2023, 16 enero). *Pantalla de cristal líquido*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Pantalla_de_cristal_l%C3%ADquido
- colaboradores de Wikipedia. (2023b, febrero 3). *Led*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Led