Proyecto Sistemas Electrónicos:
Diseño de un sistema de Control de Accesos a Edificio

Javier Grande Alzola
Jaime Garrido González

Enero 2025

1. Descripción del proyecto

El proyecto consiste en el diseño de un sistema de control de acceso a un edificio. Podrán acceder 16 usuarios, formados por el usuario 0, que será el manager, y los otros 15. Cada uno de ellos tiene un nombre de usuario de 6 caracteres, además de un PIN de 4 caracteres que deberán introducir para acceder al edificio. Si el usuario introduce el PIN de forma incorrecta más de 3 veces, se bloqueará y solo el manager podrá desbloquearlo. Los usuarios podrán estar en 3 situaciones diferentes:

- Trabajando: Al entrar al edificio, los usuarios empezarán a trabajar. En este estado, los usuarios solo pueden pasar a realizar una pausa o salir del edificio.
- Pausa: En este estado, los usuarios harán una pausa (ir al cuarto de baño, a tomar café, etc...)
- Salir: En este estado, los usuarios salen del edificio y termina su turno de trabajo.

En cada uno de los estados, se informará a la central de la actividad de los usuarios, incluyendo la hora, a qué estado pasan, etc...

Por otra parte, el manager podrá desbloquear los usuarios, o pedir un informe de cada uno de ellos, en el que aparecerá el estado, tiempo total trabajado, etc...

2. Desarrollo

Para llevar a cabo el proyecto, hemos diseñado 2 programas principales:

- Escritura en FLASH: En este programa, escribimos los nombres de cada usuario, almacenados en un char de 6 caracteres, además de sus respectivos pines de 4 caracteres. En total tenemos 160 caracteres.
- Sistema principal: Es el programa principal, en el que desarrollamos el proceso.

2.1. Escritura en FLASH

Como se ha comentado anteriormente, en este programa cargamos los datos en la FLASH. Definimos primero los parámetros y las cadenas de tipo "char" en las que almacenaremos los nombres y los PINES:

```
// Definición de parametros
#define MAX_USERS 16
#define NAME_LENGTH 6
#define PASSWORD_LENGTH 4

// Nombres de los usuarios
char user_names[MAX_USERS][NAME_LENGTH] =
{ "0JLKOP", "1RZQMA", "2LVCOW", "3PZMRT", "4WNVKS",
  "5KJRQE", "6VCNZR", "7QXJLO", "8QSVTM", "9BNQWR",
  "10LJPT", "11QMVS", "12ZWPL", "13VTNM", "14WZXC", "15KJDA" };

// Contraseñas de los usuarios (4 caracteres por usuario)
char user_passwords[MAX_USERS][PASSWORD_LENGTH] =
{ "4821", "7194", "5638", "2347", "8412",
  "1926", "5703", "8039","1145", "7620",
  "2184", "9072", "3371", "6249", "1185", "4269" };
```

Posteriormente, definimos y configuramos los siguientes elementos:

- Funciones a usar.
- Desactivamos Watch-Dog.
- Timer de 6 ms.

```
// Definición de funciones
void conf_reloj(char VEL);
void guarda_flash(char *dato, char *address, int length);

int main(void)
{
    //Desactivamos Watchdog
    WDTCTL = WDTPW | WDTHOLD;
    conf_reloj(1);

    /------ Configuración de la USCI-A para modo SPI:----------/
    UCB0CTL1 |= UCSWRST; // Resetea la USCI
    UCB0CTL0 = UCCKPH | UCMSB | UCMST | UCMODE_0 | UCSYNC;
    UCB0CTL1 = UCSSEL_2 | UCSWRST;
    UCB0BR0 = 2;
```

```
    UCB0CTL1 &= ~UCSWRST;

    //Timer de 1 ms
    TA0CCTL0 = CM_0 | CCIS_0 | OUTMOD_0 | CCIE;
    TA0CTL = TASSEL_1 | ID_0 | MC_1;
    TA0CCR0 = 11999;

    __bis_SR_register(GIE);
```

Llamamos a la función guarda flash, apuntando a la dirección 0x1000, donde comenzará a escribir los caracteres cada usuario y a continuación, escribimos las contraseñas a partir de la 0x1060.

```
// Llamada a la función guarda_flash
    guarda_flash((char )user_names, (char) 0x1000,
                    MAX_USERS * NAME_LENGTH);
    guarda_flash((char )user_passwords, (char) 0x1060,
                    MAX_USERS * PASSWORD_LENGTH);
    while (1)  __bis_SR_register(LPM3_bits);
```

Por último, diseñamos la función guardaflash, en la que distinguimos 2 situaciones:

- Si la dirección a la que se apunta está vacia, entonces se escriben los datos a partir de la dirección elegida.
- Si la dirección a la que se apunta está escrita, se borra y posteriormente se escribe el dato.

```
//Función en la que escribimos los datos en la FLASH
void guarda_flash(char *dato, char *address, int length)
{
    unsigned int i;
    for (i = 0; i < length; i++)
    {
        // Si la dirección está vacia, escribimos el dato contenido
        // en la dirección a la que se apunta
        if (address[i] == 0xFF)
        {
            FCTL3 = FWKEY;
            FCTL1 = FWKEY + WRT;
            *(address+i) = *(dato+i);
            FCTL1 = FWKEY;
            FCTL3 = FWKEY + LOCK;
        }
```

```
        //Si no está vacia, borramos el dato y escribimos el deseado
        else
        {
            FCTL1 = FWKEY + ERASE;
            FCTL3 = FWKEY;
            *(address+i) = 0xFF;
            FCTL1 = FWKEY + WRT;
            *(address+i) = *(dato+i);
            FCTL1 = FWKEY;
            FCTL3 = FWKEY + LOCK;
        }
    }
}
```

Para finalizar, declaramos el vector de interrupciones y la función confreloj:

```
#pragma vector=TIMER0_A0_VECTOR
__interrupt void TIMER0_A0_ISR_HOOK(void)
{__bic_SR_register_on_exit(LPM3_bits);}

void conf_reloj(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;}
```

Una vez ejecutado este programa, cuyo objetivo es la escritura en la FLASH del microcontrolador MSP430, continuaremos con el programa principal, denominado Sistema principal. La escritura en memoria debe quedar de la siguiente forma:

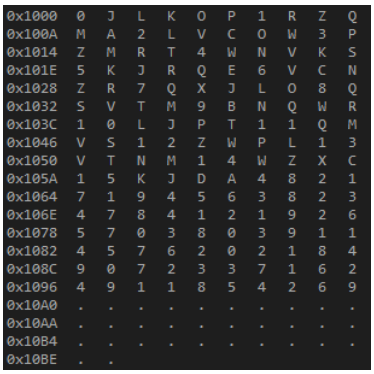

Figura 1: Memoria FLASH

2.2. Sistema principal

Inicialmente, comenzamos declarando las bibliotecas, necesarias para la ejecución del código:

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

Declaramos también las funciones a usar:

```
//Funciones
//-------------------------------------------------------------
void Set_Clk(char VEL);
void UART_SendString(const char *str);
void UARTprintCR(const char *frase);
int lee_ch(char canal);
void inicia_ADC(char canales);
int leer_ADC (int canal, int eje, int valor, int maxeje,
              int mineje, int maxvalor, int minvalor, int aumento);
```

Posteriormente, las variables que usaremos a lo largo del código:

```
//Variables
//-------------------------------------------------------------------
char * LeeFlash;
enum estados {ARRANQUE, SELEC_USER, PIN_US, MODO, MODO_ADMIN,
              INFORME, START_SELEC_USER, };
char estados = ARRANQUE;
volatile char buffer[3] = { 0 }, PIN_in[5] = { 0 };
volatile int dia = 0, mes = 0, ano = 0, hora = 0, min = 0, seg = 0;
int j = 0, l = 0, intentos = 3, ejey = 512, y = 45, ya = 0;
char usuario[7], usuario_ant[7], acceso[90], PIN[4], fecha[15];

//0:Fuera del edificio
//1:Trabajando
//2:Pausa
//3:Bloqueo

unsigned int modo_ant[16] = { 0, 0, 0, 0, 0, 0, 0, 0, 0,
                              0, 0, 0, 0, 0, 0, 0 };
unsigned int modos[16] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
                              0, 0, 0, 0, 0, 0 };
int tiempos[16] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                    0, 0, 0, 0 };
int tiempo_p[16] ={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                   0, 0, 0, 0};
volatile int fin = 0, tms = 0, fin_fecha = 0;
unsigned int pin_correcto = 0, t = 0, id = 0, i = 0, p = 0;

 Graphics_Context g_sContext;
```

Declaramos ahora los siguientes elementos:

- Funciones a usar.
- Desactivamos Watch-Dog.
- Pines para los LEDs del BoosterPack.
- Pines del botón del Joystick.
- Inicialización de la UART.
- Temporizador de 10 ms.

A su vez, también se imprime el mensaje inicial, en el que se pide al usuario que introduzca por puerto serie la fecha y hora.

```
int main(void)
 {
     WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer


     BCSCTL2 = SELM_0 | DIVM_0 | DIVS_0;

     if (CALBC1_16MHZ != 0xFF) {
         __delay_cycles(100000);
         DCOCTL = 0x00;
         BCSCTL1 = CALBC1_16MHZ;     /* Set DCO to 16MHz */
         DCOCTL = CALDCO_16MHZ;}

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

     //Declaramos los LEDs del BoosterPack
     P2DIR |= BIT1 | BIT2 | BIT3 |BIT4;
     P2OUT &= ~BIT1;
     P2OUT &= ~BIT2;
```

```
21      P2OUT &= ~BIT3;
22      P2OUT &= ~BIT4;
23
24
25      inicia_ADC(BIT0+BIT3);
26
27      /------ Pines de E/S involucrados:-----------------------/
28      P1SEL2 = BIT1 | BIT2;  //P1.1 RX, P1.2: TX
29      P1SEL = BIT1 | BIT2;
30      P1DIR = BIT0 + BIT2;
31      P1REN = BIT3;              //Boton en P1.3
32      P1OUT = BIT3;
33
34      //Declaro el boton del joystick
35      P2DIR &= ~BIT5;
36      P2OUT |= BIT5;
37
38      /------ Configuraci n de la USCI-A para modo UART:---------/
39
40      UCA0CTL1 |= UCSWRST; // Reset
41      UCA0CTL1 = UCSSEL_2 | UCSWRST; //UCSSEL_2: SMCLK (16MHz)
42      UCA0CTL0 = 0; // 8bit, 1stop, sin paridad. NO NECESARIO
43      UCA0BR0 = 139; // 16MHz/139=115108... //Velocidad
44      UCA0CTL1 &= ~UCSWRST; /* Quita reset */
45      IFG2 &= ~(UCA0RXIFG); /* Quita flag */
46      IE2 |= UCA0RXIE; /* y habilita int. recepcion */
47
48      //Temporizador de 10 ms
49      TA1CTL = TASSEL_2 | ID_3 | MC_1;//SMCLK, DIV=8 (2MHz), UP
50      TA1CCR0 = 19999;       //periodo=20000: 10ms
51      TA1CCTL0 = CCIE;      //CCIE=1
52
53      //Inicialización de la pantalla
54      Crystalfontz128x128_Init();
55      Crystalfontz128x128_SetOrientation(LCD_ORIENTATION_UP);
56      Graphics_initContext(&g_sContext, &g_sCrystalfontz128x128);
57      Graphics_setFont(&g_sContext, &g_sFontCm16b);
58      Graphics_clearDisplay(&g_sContext);
59      Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_WHITE);
60      Graphics_setBackgroundColor(&g_sContext, GRAPHICS_COLOR_BLACK);
61
62      UART_SendString("Introduzca la fecha y
63                    hora en formato DD/MM/AA HH:MM\r\n");
64
65      // Mensaje por pantalla
```

```
66      Graphics_drawString(&g_sContext, "Introduzca la",
67                          13, 10, 40, TRANSPARENT_TEXT);
68      Graphics_drawString(&g_sContext, "fecha y hora",
69                          13, 10, 60, TRANSPARENT_TEXT);
70
71      __bis_SR_register(GIE);
```

Figura 2: Pantalla inicial

Al comenzar con while, configuramos la actualización de la fecha y la actualización de tiempos en función del modo en el que se encuentre el usuario:

```
1   while (1)
2     {
3         LPM0; //Bucle de bajo consumo
4
5         if(fin_fecha == 1)
6         {
7             if (seg >= 60)
8             {
9                 Graphics_setForegroundColor(&g_sContext,
10                                            GRAPHICS_COLOR_BLACK);
11                sprintf(fecha, "%d/%d/%d  %d:%d",
12                        dia, mes, ano, hora, min);
13                Graphics_drawString(&g_sContext, fecha,
14                                    20, 15, 5,  TRANSPARENT_TEXT);
15
16                for (p = 0; p < 15 ; p++)
17                {
18                    if (modos[p] == 1)
19                    {
20                        tiempos[p]++;
21                    }
```

```
22
23                      if (modos[p] == 0)
24                      {
25                          tiempos[p] = 0;
26                          tiempo_p[p] = 0;
27                      }
28
29                      if (modos[p] == 2)
30                      {
31                          tiempo_p[p]++;
32                      }
33                  }
34
35                  min++;
36                  seg = 0;
37
38                  if(min>=60)
39                  {
40                      hora++;
41                      min = 0;
42                  }
43                  if(hora>=24)
44                  {
45                      dia++;
46                      hora = 0;
47                  }
48
49
50                  Graphics_setForegroundColor(&g_sContext,
51                                              GRAPHICS_COLOR_WHITE);
52                  sprintf(fecha, "%d/%d/%d  %d:%d", dia, mes, ano,
53                          hora, min);
54                  Graphics_drawString(&g_sContext, fecha, 20, 15, 5,
55                                      TRANSPARENT_TEXT);
56              }
57          }
58      }
59  }
```

Posteriormente, describimos el primer estado, en el que comenzamos inicialmente. Para ejecutar el bloque, se debe cumplir la condición de que fin sea igual a 1, que se cumplirá al introducir la fecha por puerto serie. Para empezar, hacemos que el LED tricolor del BoosterPack parpadee en ámbar. Una vez introducida la fecha y si es válida, escribiremos un mensaje por puerto serie y se mostrará la fecha por pantalla, que comenzará a actualizarse al asignar a finfecha el valor 1, ejecutándose el código superior. En caso de no ser válida, escribiremos un mensaje al usuario para que vuelva a introducirla. Además, si pasan 10 segundos y no hemos introducido un dato, imprimiremos un mensaje de espera por puerto serie.

```
switch (estados)
    {
    //Estado de arranque: introducimos la hora y pasamos a seleccionar usuario

    case ARRANQUE:
        //Parpadeo de LED ambar
        if (l >= 50)
        {
            P2OUT ^= (BIT1 | BIT2);
            l = 0;
        }
        //Variable que se pone a 1 al finalizar
        //la introducción de la fecha
        if (fin == 1)
        {
            //Borro mensaje anterior
            Graphics_setForegroundColor(&g_sContext,
                                        GRAPHICS_COLOR_BLACK);
            Graphics_drawString(&g_sContext, "Introduzca la",
                                13, 10, 40,  TRANSPARENT_TEXT);
            Graphics_drawString(&g_sContext, "fecha y hora",
                                13, 10, 60, TRANSPARENT_TEXT);
            //Validación de fecha
            if (( dia > 31) || (mes > 12) ||
                (hora > 23) || (min > 59))
            {
                fin = 0;
                UART_SendString("Error, introduzcala de nuevo \r\n");
                Graphics_setForegroundColor(&g_sContext,
                                            GRAPHICS_COLOR_WHITE);
                Graphics_drawString(&g_sContext, "Introduzca la",
                                    13, 10, 40, TRANSPARENT_TEXT);
                Graphics_drawString(&g_sContext, "fecha y hora",
                                    13, 10, 60, TRANSPARENT_TEXT);
```

```
                fin_fecha = 2;
            }
            else
            {
                Graphics_setForegroundColor(&g_sContext,
                                            GRAPHICS_COLOR_WHITE);
                sprintf(fecha, "%d/%d/%d  %d:%d",
                        dia, mes, ano, hora, min);
                Graphics_drawString(&g_sContext, fecha,
                                    20, 15, 5, TRANSPARENT_TEXT);
                UART_SendString("Fecha y horas introducidas
                                con exito \r\n");
                estados = SELEC_USER;
                fin_fecha = 1;
            }
        }
        else
        {
            if(t>=1000)
            {
                UART_SendString("Esperando \r\n");
                t=0;
            }
            break;
        }
        break;
```

Pasamos al estado de selección de usuario, en el que pasamos el LED a color azul y mediante una función leemos el valor del Joystick, moviéndonos así entre usuarios.

Para leer el usuario de la FLASH, hemos usado un puntero "LeeFlash"que inicializamos en la posición del primer caracter del primer usuario. Usando un búcle for, recorremos los 6 caracteres guardándolos así en una variable para almacenar el usuario anterior, más el terminador de cadena. Posteriormente actualizamos los índices j e id para movernos entre los usuarios al leer la memoria FLASH haciendo uso de la función leer ADC. Al actualizar dichos índices, guardamos el siguiente usuario en otra variable y borramos la anterior, para pintar después en blanco el siguiente.

Si el usuario está trabajando (1), puede elegir entre hacer una pausa o salir.

```
//Estado de seleccion de usuario

```

```
case SELEC_USER:
    //Encendemos el LED azul
    P2OUT |= BIT4;
    P2OUT &= ~(BIT1 | BIT2 | BIT3)
    //Pedimos seleccion de usuario
    Graphics_setForegroundColor(&g_sContext,
                                GRAPHICS_COLOR_WHITE);
    Graphics_drawString(&g_sContext, "Seleccione ", 13, 10, 30,
                        TRANSPARENT_TEXT);
    Graphics_drawString(&g_sContext, "usuario ", 13, 10, 50,
                        TRANSPARENT_TEXT);
    LeeFlash= (char *) 0x1000;
    if (t >= 50)
    {
        for (i = 0; i <= 5; i++)  usuario_ant[i] = LeeFlash[i+j];
        usuario_ant[7] = '\0'
        j = leer_ADC (3, ejey, j, 800, 223, 90, 0, 6);
        id = leer_ADC(3, ejey, id, 800, 223, 15, 0 , 1);
        // Copiamos el nuevo nombre desde la memoria Flash
        for (i = 0; i <= 5; i++)  usuario[i] = LeeFlash[i + j];
        usuario[7] = '\0';
        // Borrar el nombre anterior
        Graphics_setForegroundColor(&g_sContext,
                                    GRAPHICS_COLOR_BLACK);
        Graphics_drawString(&g_sContext, usuario_ant, 6, 35, 100,
                            TRANSPARENT_TEXT);
        // Dibujamos el nuevo nombre
        Graphics_setForegroundColor(&g_sContext,
                                    GRAPHICS_COLOR_WHITE);
        Graphics_drawString(&g_sContext, usuario, 6, 35, 100,
                            TRANSPARENT_TEXT);
        t = 0;  // Reiniciar el temporizador
    }
    if(!(P2IN&BIT5))
    {
        if(t>=20)
        {
            Graphics_setForegroundColor(&g_sContext,
                                        GRAPHICS_COLOR_BLACK);
            Graphics_drawString(&g_sContext, "Seleccione ", 13, 10,
                                30,TRANSPARENT_TEXT);
            Graphics_drawString(&g_sContext, "usuario ", 13, 10, 50,
                                TRANSPARENT_TEXT);
            fin=2;
            intentos=3;
```

```
48              estados = PIN_US;
49              t=0;
50          }
51      }
52      break
```

Figura 3: Selección de usuario

Posteriormente, pasamos al estado PIN US, en el que se introduce el PIN del usuario seleccionado

```
1  //Estado en el que se introduce el PIN del usuario
2
3  case PIN_US:
4      P2OUT|= (BIT3 | BIT1);
5      if(intentos<3 && t>=100)
6      {
7          Graphics_setForegroundColor(&g_sContext,
8                                      GRAPHICS_COLOR_BLACK);
9          Graphics_drawString(&g_sContext, "ERROR", 13, 30, 30,
10                             TRANSPARENT_TEXT);
11     }
12     if(intentos == 0 && t>=400)
13     {
14         Graphics_setForegroundColor(&g_sContext,
15                                     GRAPHICS_COLOR_BLACK);
16         Graphics_drawString(&g_sContext, "Usuario", 13, 15, 50,
17                             TRANSPARENT_TEXT);
18         Graphics_drawString(&g_sContext, "bloqueado", 13, 10, 70,
19                             TRANSPARENT_TEXT);
20         t=0;
```

```
        estados = START_SELEC_USER;
    }
    LeeFlash = (char *)0x1060
    for (i = 0; i < 4; i++) PIN[i] = LeeFlash[i+4*id]
    if(fin==2)
    {
        Graphics_setForegroundColor(&g_sContext,
                                    GRAPHICS_COLOR_WHITE);
        Graphics_drawString(&g_sContext, "Elija el PIN",
                            13, 10, 30, TRANSPARENT_TEXT);
        fin = 3;
    }
    if(fin == 4)
    {
        for (i = 0; i < 4; i++) if (PIN_in[i] == PIN[i])
        {
        pin_correcto++;
        }
        for (i = 0; i < 4; i++) PIN_in[i] = '0'
        if (pin_correcto == 4)
        {
            Graphics_setForegroundColor(&g_sContext,
            GRAPHICS_COLOR_BLACK);
            Graphics_drawString(&g_sContext, "Elija el PIN",
                                13, 10, 30, TRANSPARENT_TEXT);
            Graphics_drawString(&g_sContext, "ERROR",
                                13, 30, 30, TRANSPARENT_TEXT);
            Graphics_drawString(&g_sContext, "Introduzca de",
                                13, 10, 50, TRANSPARENT_TEXT);
            Graphics_drawString(&g_sContext, "nuevo el PIN",
                                13, 15, 70, TRANSPARENT_TEXT)
            Graphics_setForegroundColor(&g_sContext,
                                        GRAPHICS_COLOR_WHITE)
            Graphics_drawString(&g_sContext, "PIN correcto ",
                                13, 10, 30,TRANSPARENT_TEXT)
            estados = MODO;
        }
        if (pin_correcto != 4)
        {
            P2OUT &= ~BIT1;
            P2OUT &= ~BIT4;
            P2OUT &= ~BIT3;
            intentos--;
            Graphics_setForegroundColor(&g_sContext,
                                        GRAPHICS_COLOR_BLACK);
```

```
66              Graphics_drawString(&g_sContext, "Elija el PIN",
67                                  13, 10, 30, TRANSPARENT_TEXT);
68              Graphics_setForegroundColor(&g_sContext,
69                                          GRAPHICS_COLOR_WHITE);
70              Graphics_drawString(&g_sContext, "ERROR",
71                                  13, 30, 30, TRANSPARENT_TEXT);
72              Graphics_drawString(&g_sContext, "Introduzca de",
73                                  13, 10, 50,TRANSPARENT_TEXT);
74              Graphics_drawString(&g_sContext, "nuevo el PIN",
75                                  13, 15, 70,TRANSPARENT_TEXT);
76              UART_SendString("Introduzca el PIN de nuevo: \r\n");
77              fin=3;
78              t=0;
79              pin_correcto = 0;
80          }
81          if (intentos == 0)
82          {
83              Graphics_setForegroundColor(&g_sContext,
84                                          GRAPHICS_COLOR_BLACK);
85              Graphics_drawString(&g_sContext, "ERROR", 13, 30, 30,
86                                          TRANSPARENT_TEXT);
87              Graphics_drawString(&g_sContext, "Introduzca de", 13, 10,
88                                  50,TRANSPARENT_TEXT);
89              Graphics_drawString(&g_sContext, "nuevo el PIN", 13, 15,
90                                  70,TRANSPARENT_TEXT);
91              pin_correcto = 0
92              if (l >= 50)
93              {
94                  P2OUT ^= BIT3;
95                  l = 0;
96
97              if(id != 0)
98              {
99                  Graphics_setForegroundColor(&g_sContext,
100                                             GRAPHICS_COLOR_WHITE);
101                 sprintf(acceso, "%s ha sido bloqueado, Hora:%d/%d/%d
102                     %d:%d \r\n", usuario, dia, mes, ano, hora, min);
103                 Graphics_drawString(&g_sContext, "Usuario", 13, 15, 50,
104                 Graphics_drawString(&g_sContext, "bloqueado", 13, 10,
105                                     70,TRANSPARENT_TEXT);
106                 UART_SendString(acceso)
107                 modos[id] = 3;
108                 LeeFlash= (char *) 0x1000;
109                 for (i = 0; i <= 5; i++)
110                 {
```

```
111                     usuario_bloq[i] = LeeFlash[i + j];
112                 }
113                 usuario_bloq[7] = '\0';
114 
115             if(id == 0)
116             {
117                 Graphics_setForegroundColor(&g_sContext,
118                                             GRAPHICS_COLOR_WHITE);
119                 Graphics_drawString(&g_sContext, "Bloqueo", 13, 15,
120                                     50,TRANSPARENT_TEXT);
121                 Graphics_drawString(&g_sContext, "total", 13, 10,
122                                     70,TRANSPARENT_TEXT);
123                 UART_SendString("Bloqueo total");
124             }
125         }
126 
127     break;
```

Figura 4: Selección de PIN

Pasamos al estado MODO, en el que distinguimos en el caso de que sea un trabajador para el que tendremos 3 estados: trabajando, pausa, salir, y para el caso de admin, en el que podremos desbloquear a los usuarios bloqueados o pedir un informe sobre la actividad de los trabajadores, en el que obtendremos el estado, tiempo trabajado y tiempo en pausa del trabajador seleccionado.

```
//En Modo seleccionamos qué pasará a hacer cada usuario
//-------------------------------------------------------------------
case MODO:
    Graphics_setForegroundColor(&g_sContext,
                                GRAPHICS_COLOR_BLACK);
    Graphics_drawString(&g_sContext, "PIN correcto ", 13, 10, 30,
                        TRANSPARENT_TEXT)
    //En caso de ser un usuario normal
    if (id != 0)
    {
        P2OUT &= ~(BIT1 | BIT4);
        P2OUT |= (BIT3 | BIT2)
        if (modos[id] != 1)
        {
            Graphics_setForegroundColor(&g_sContext,
                                        GRAPHICS_COLOR_WHITE);
            Graphics_drawString(&g_sContext, "ENTRAR", 13, 25, 55,
                                TRANSPARENT_TEXT);
            Graphics_fillCircle(&g_sContext, 15, 60, 5)
            if (!(P2IN & BIT5) && modos[id]==0)
            {
                sprintf(acceso, "%s ha entrado, Hora:%d/%d/%d  %d:%d,
                Intento:%d \r\n", usuario, dia, mes, ano, hora, min, 3-intentos);
                UART_SendString(acceso)
                modos[id] = 1;
                modo_ant[id] = modos[id];
                estados = REINICIO;

            if (!(P2IN & BIT5) && modos[id]==2)
            {
                sprintf(acceso, "%s ha entrado, Hora:%d/%d/%d  %d:%d,
                Tiempo trabajado: %d min, Tiempo en pausa: %d min \r\n",
                usuario, dia, mes, ano, hora, min, tiempos[id], tiempo_p[id]);
                UART_SendString(acceso);
                modos[id] = 1;
                modo_ant[id] = modos[id];
                estados = REINICIO;
            }
            break;
```

```
        if (modos[id] == 1)
        {
            Graphics_setForegroundColor(&g_sContext,
                                        GRAPHICS_COLOR_WHITE);
            Graphics_drawString(&g_sContext, "PAUSA", 13, 25, 40,
                                TRANSPARENT_TEXT);
            Graphics_drawString(&g_sContext, "SALIR", 13, 25, 70,
                                TRANSPARENT_TEXT);
            Graphics_fillCircle(&g_sContext, 15, y, 5)
            if (t >= 50)
            {
                y = leer_ADC (3, ejey, y, 800, 223, 75, 45, 30);
                if (y != ya)
                {
                    Graphics_setForegroundColor(&g_sContext,
                                                GRAPHICS_COLOR_BLACK);
                    Graphics_fillCircle(&g_sContext, 15, ya, 5);
                    Graphics_setForegroundColor(&g_sContext,
                                                GRAPHICS_COLOR_WHITE);
                    Graphics_fillCircle(&g_sContext, 15, y, 5);
                    ya = y;
                }
                t = 0;
            }
            if (!(P2IN & BIT5))
            {
                if (y == 75)

                    sprintf(acceso, "%s ha salido, Hora:%d/%d/%d  %d:%d,
                    Tiempo trabajado: %d min  \r\n", usuario, dia, mes, ano,
                    hora, min, tiempos[id]);
                    UART_SendString(acceso);
                    modos[id] = 0;
                    modo_ant[id] = modos[id];
                }
                if (y == 45)
                {
                    sprintf(acceso, "%s ha ido a hacer una pausa,
                    Hora:%d/%d/%d  %d:%d \r\n", usuario, dia, mes, ano, hora, min);
                    UART_SendString(acceso);
                    modos[id] = 2;
                    modo_ant[id] = modos[id];
                }
                estados = REINICIO;
```

```
            }
            break;
        }
    }
```

(a) Estado 0 (b) Estado 1

Figura 5: Imágenes lado a lado.

```
//Si es el admin
if(id==0)
{
     P2OUT |= (BIT3 | BIT2 | BIT4 | BIT1);
     if (l >= 100)
     {
         P2OUT ^= (BIT3 | BIT2 | BIT4 | BIT1);
         l = 0;
     }
    Graphics_setForegroundColor(&g_sContext,
                                GRAPHICS_COLOR_WHITE);
    Graphics_drawString(&g_sContext, "DESBLOQUEO", 13, 25, 40,
                        TRANSPARENT_TEXT);
    Graphics_drawString(&g_sContext, "INFORME", 13, 25, 70,
                        TRANSPARENT_TEXT);
    Graphics_fillCircle(&g_sContext, 15, y, 5);
    if (t >= 50)
    {
        y = leer_ADC (3, ejey, y, 800, 223, 75, 45, 30);
        if (y != ya)
        {
            Graphics_setForegroundColor(&g_sContext,
                                        GRAPHICS_COLOR_BLACK);
            Graphics_fillCircle(&g_sContext, 15, ya, 5);
            Graphics_setForegroundColor(&g_sContext,
```

```
                                                GRAPHICS_COLOR_WHITE);
            Graphics_fillCircle(&g_sContext, 15, y, 5);
            ya = y;
        }
        t = 0;
    }
    if (!(P2IN&BIT5))
    {
        Graphics_setForegroundColor(&g_sContext,
                                    GRAPHICS_COLOR_BLACK);
        Graphics_drawString(&g_sContext, "DESBLOQUEO", 13, 25, 40,
                            TRANSPARENT_TEXT);
        Graphics_drawString(&g_sContext, "INFORME", 13, 25, 70,
                            TRANSPARENT_TEXT);
        Graphics_fillCircle(&g_sContext, 15, y, 5);
        if (y == 75)
        {
            UART_SendString( "El manager ha pedido un informe\r\n");
            estados = INFORME;
        }
        if (y == 45)
        {
            if (modos[i] == 3)
            {
                UART_SendString("Los usuarios han sido
                                desbloqueados\r\n");
            }
            for(i = 0; i < 16; i ++) modos[i] = modo_ant[i];
            estados = REINICIO;
        }
    }
}
break;
```

Para el caso en el que el manager pida un informe, vamos a un estado distinto, en el que seleccionamos el usuario con el josytick e imprimimos por terminal el estado en el que se encuentra, el tiempo trabajado y el tiempo en pausa. Para distinguir los estados, emplearemos los siguientes índices:

- 0: Usuario fuera del edificio.
- 1: Usuario trabajando.
- 2: Usuario en pausa.

Figura 6: Modo Manager

```
//Informe: se selecciona el usuario y se muestra el estado y
//tiempo trabajado por pantalla
//-----------------------------------------------------------------
case INFORME:
    LeeFlash= (char *) 0x1000;
    //Pedimos seleccion de usuario
    Graphics_setForegroundColor(&g_sContext,
                                GRAPHICS_COLOR_WHITE);
    Graphics_drawString(&g_sContext, "Seleccione ", 13, 10, 30,
                        TRANSPARENT_TEXT);
    Graphics_drawString(&g_sContext, "usuario para ", 13, 10, 50,
                        TRANSPARENT_TEXT);
    Graphics_drawString(&g_sContext, "informe ", 13, 10, 70,
                        TRANSPARENT_TEXT)
    if (t >= 50) {
        for (i = 0; i <= 5; i++)  usuario_ant[i] = LeeFlash[i+j];
        usuario_ant[7] = '\0'
        j = leer_ADC (3, ejey, j, 800, 223, 90, 0, 6);
        id = leer_ADC(3, ejey, id, 800, 223, 15, 0 , 1);
        // Copiar el nuevo nombre desde la memoria Flash
        for (i = 0; i <= 5; i++)  usuario[i] = LeeFlash[i + j];
        usuario[7] = '\0';
        // Borrar el nombre anterior
        Graphics_setForegroundColor(&g_sContext,
                                    GRAPHICS_COLOR_BLACK);
        Graphics_drawString(&g_sContext, usuario_ant, 6, 35, 100,
```

```
                                TRANSPARENT_TEXT);
        // Dibujar el nuevo nombre (en blanco)
        Graphics_setForegroundColor(&g_sContext,
                                    GRAPHICS_COLOR_WHITE);
        Graphics_drawString(&g_sContext, usuario, 6, 35, 100,
                                TRANSPARENT_TEXT);
        t = 0;  // Reiniciar el temporizador

    if(!(P2IN&BIT5))
    {
        if(t>=20)
        {
            Graphics_setForegroundColor(&g_sContext,
                                        GRAPHICS_COLOR_BLACK);
            Graphics_drawString(&g_sContext, "Seleccione ",
                                13, 10, 30,TRANSPARENT_TEXT);
            Graphics_drawString(&g_sContext, "usuario para ",
                                13, 10, 50,TRANSPARENT_TEXT);
            Graphics_drawString(&g_sContext, "informe ",
                                13, 10, 70, TRANSPARENT_TEXT)
            sprintf(acceso, "%s ,Estado:%d, Tiempo trabajado:
                    %d min, Tiempo en pausa: %d  \r\n",
                    usuario, modos[id], tiempos[id],   tiempo_p[id]);
            UART_SendString(acceso);
            estados = REINICIO;
            t = 0;
        }

    break;
```

Para finalizar con la máquina de estados, el programa pasa a START SELEC USER, estado en el que se reinician las variables y se eliminan los elementos de selección por pantalla.

```
//Volvemos a seleccion de usuario
case REINICIO:
    t=0;
    id=0;
    j=0;
    pin_correcto=0;
    Graphics_setForegroundColor(&g_sContext,
                                GRAPHICS_COLOR_BLACK);
    Graphics_drawString(&g_sContext, "ENTRAR", 13, 25, 55,
                        TRANSPARENT_TEXT);
```

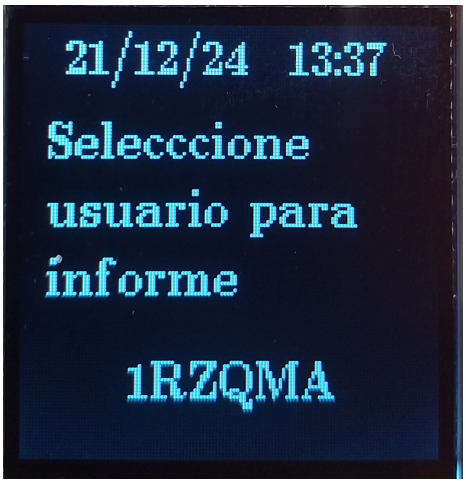

Figura 7: Informe

```
    Graphics_drawString(&g_sContext, usuario, 6, 35, 100,
                        TRANSPARENT_TEXT);
    Graphics_drawString(&g_sContext, "SALIR", 13, 25, 70,
                        TRANSPARENT_TEXT);
    Graphics_drawString(&g_sContext, "PAUSA", 13, 25, 40,
                        TRANSPARENT_TEXT);
    Graphics_fillCircle(&g_sContext, 15, y, 5);
    Graphics_fillCircle(&g_sContext, 15, 60, 5);
    estados=SELEC_USER;
}
```

Para poder obtener los valores al escribirlos por el terminal, como la fecha inicial para arrancar el sistema y el PIN de cada usuario, usamos el siguiente código en la parte de interrupciones:

```
/---- Interrupciones ----/
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR_HOOK(void)
{

    static int index = 0;
    static int num_dato = 0;

    if (num_dato == 0)
    {
        buffer[index] = UCA0RXBUF;
        index++;
```

```
13          if (index >= 2)
14          {
15              buffer[2] = '\0';
16              dia = atoi(buffer);
17              num_dato = 1;
18              index = 0;
19          }
20
21      }
22
23      else if (num_dato == 1)
24      {
25          buffer[index] = UCA0RXBUF;
26          index++;
27          if (index >= 2)
28          {
29              buffer[2] = '\0';
30              mes = atoi(buffer);
31              num_dato = 2;
32              index = 0;
33          }
34
35      }
36
37      else if (num_dato == 2)
38      {
39          buffer[index] = UCA0RXBUF;
40          index++;
41          if (index >= 2)
42          {
43              buffer[2] = '\0';
44              ano = atoi(buffer);
45              num_dato = 3;
46              index = 0;
47
48          }
49      }
50
51      else if (num_dato == 3)
52      {
53          buffer[index] = UCA0RXBUF;
54          index++;
55          if (index >= 2)
56          {
57              buffer[2] = '\0';
```

```
            hora = atoi(buffer);
            num_dato = 4;
            index = 0;

        }
    }

    else if (num_dato == 4)
    {
        buffer[index] = UCA0RXBUF;
        index++;
        if (index >= 2)
        {
            buffer[2] = '\0';
            min = atoi(buffer);
            num_dato = 5;
            index = 0;
            fin = 1;
        }
    }

    if (fin == 3)
    {
        PIN_in[index] = UCA0RXBUF;
        index++;
        if (index >= 4)
        {
            PIN_in[4] = '\0';
            fin = 4;
            index = 0;
        }
    }
    LPM0_EXIT;
}
```

Además, usamos las siguientes interrupciones para el ADC y para llevar a cabo acciones en el código:

```
#pragma vector=ADC10_VECTOR
__interrupt void ConvertidorAD(void)
{
    LPM0_EXIT;
}

#pragma vector=TIMER1_A0_VECTOR
```

```
__interrupt void Interrupcion_T1(void)
{

    l++; // Usada para el parpadeo de los LEDs
    t++; // Usada para coordinar las acciones en el programa
    seg++; //Usada para actualizar el tiempo del sistema
    LPM0_EXIT;

}
```

Por último, describimos las funciones usadas en el código:

- Funcion leer ADC: La usamos para asignar valores a i e id, índices para recorrer la memoria FLASH. Almacenamos en la variable eje el valor del joystick y establecemos los límites.

```
int leer_ADC (int canal, int eje, int valor, int maxeje,
int mineje, int maxvalor, int minvalor, int aumento)
{
    eje = lee_ch(canal);
    if (eje <= maxeje) valor += aumento;
    if (eje >= mineje) valor -= aumento;
    if (valor > maxvalor) valor = minvalor;
    if (valor < minvalor) valor = maxvalor;
    return valor;
}
```

- Función UART SendString: Usada para obtener los caracteres por puerto serie.

```
void UART_SendString(const char *str)
{
    while (*str != '\0')
    {
        while (!(IFG2 & UCA0TXIFG))
            ; // Espera a que el buffer est  listo
        UCA0TXBUF = *str; // Env a el car cter actual
        str++; // Avanza al siguiente car cter
    }
}
```

- Función inicia ADC: convertimos los valores analógicos del joystick a digitales

```
void inicia_ADC(char canales)
{
    ADC10CTL0 &= ~ENC;       //deshabilita ADC
    ADC10CTL0 = ADC10ON | ADC10SHT_3 | SREF_0 | ADC10IE;
    ADC10CTL1 = CONSEQ_0 | ADC10SSEL_0 | ADC10DIV_0 |
                SHS_0 | INCH_0;
    //Modo simple, reloj ADC, sin subdivision,
    //Disparo soft, Canal 0
    ADC10AE0 = canales; //habilita los canales indicados
    ADC10CTL0 |= ENC; //Habilita el ADC
}
```

- Función lee ch: Leemos los valores analógicos del joystick

```
    int lee_ch(char canal)
{
    ADC10CTL0 &= ~ENC; //deshabilita el ADC
    ADC10CTL1 &= (0x0fff); //Borra canal anterior
    ADC10CTL1 |= canal << 12; //selecciona nuevo canal
    ADC10CTL0 |= ENC; //Habilita el ADC
    ADC10CTL0 |= ADC10SC; //Empieza la conversi n
    LPM0; //Espera fin en modo LPM0
    return (ADC10MEM); //Devuelve valor leido
}
```

3. Recursos

Para llevar a cabo el proyecto, hemos usado 15896 bytes y 425 bytes de RAM

4. Gestión del trabajo

- Javier: Escritura de los usuarios y sus correspondientes pines en FLASH. Desarrollo del programa principal. Parte de la memoria.
- Jaime: Desarrollo del programa principal. Parte de la memoria. Presentación.